3D Video Coding : A Novel Approach

By

Kausik Maiti

DEPARTMENT OF ELECTRICAL ENGINEERING
Indian Institute of Technology Kanpur
APRIL, 2003

3D VIDEO CODING : A NOVEL APPROACH

BY

Kausik Maiti

DEPARTMENT OF ELECTRICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY, KANPUR

APRIL, 2003

3D Video Coding : A Novel Approach

A Thesis Submitted

in Partial Fulfillment of the Requirements

for the Degree of

Master of Technology

by

Kausik Maiti

to the

DEPARTMENT OF ELECTRICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY, KANPUR

April, 2003

2 - AUG 2003
पुरुषोत्तम काशीनाथ केलकर पुस्तकालय
भारतीय प्रौद्योगिकी संस्थान कानपुर
अवाप्ति क्र० A 144427

A144427

ON 28-4-03
I.I.T., K.

CERTIFICATE

This is to certify that the work contained in the thesis entitled "*3D Video Coding : A Novel Approach*", by *Kausik Maiti*, has been carried out under my supervision and that this work has not been submitted elsewhere for the award of a degree.

(Dr. Sumana Gupta)

April 2003

Professor,
Department of Electrical Engineering,
Indian Institute of Technology,
Kanpur-208016.

ABSTRACT

The main objective of this thesis-work is to apply transform coding techniques to code video in a manner that helps to achieve improved rate-distortion performance and efficient bit-rate-control. At first, we have made combined use of DCT and sub-band coding (DWT) techniques to reach this goal. No motion compensation is used at all. For videos with small object motion (as is the case in video-phone / video-conferencing type applications), the proposed method preserves the subjective quality even at very low bit-rate (at around 0.04bpp). However, when the object-motion increases, significant artifacts have been observed in the regions containing large motion. The same has been noticed while coding video at an extreme low bit rate (around 0.02 bpp). To reduce this effect, we have proposed a modification to the algorithm. Next, we have tried to generalize the coding scheme so as to code wide variety of videos (e.g. videos with complex motions of objects, synthetic video etc.) with acceptable performance at different bit-rates. With this in view, we have replaced DCT with a set of KL transformation matrices. Through comparative study we have shown that the second method yields better results for general class of video. The price paid in return is additional complexity at the encoder and hence greater encoding delay. Finally, we have discussed how the proposed method can include interesting features like SNR-scalability, spatial-scalability, temporal-scalability and produce an embedded bit-stream that is suitable for progressive transmission over error-prone channel.

Dedicated To Him Who Is Our
Innermost Self

Acknowledgement

Foremost, I would like to express my sincere gratitude to my respected guide Dr. Sumana Gupta for her kind guidance and invaluable suggestions that have helped this thesis-work to assume a meaningful shape. I am grateful to her for providing me with useful references, materials and books that enriched my knowledge and understanding about the problem in hand. A lot many thanks too for giving me full freedom to think and approach in my own way and finally for carefully editing this thesis report. Without her active cooperation, it would not have been possible for me to complete this work in time. I am also thankful to her for her precious advices in my personal life.

I am deeply indebted to the faculty members who have taught me various Signal Processing & Communication subjects, and clarified my doubts regarding the basics, and also helped in developing my own viewpoint.

I am thankful to Dr. Amir Said and Dr. William Pearlman for their SPIHT tool that I have used in my work. Special thanks goes to Dr. Michelle Effros for providing me with her papers on Image Compression.

In the course of my thesis work, a number of other persons, too, stretched their helpful hands to me, whenever I was in need. Debi Prosad always allowed me to use his personal computer at any time, day or night, without any objection. Discussion with Gopal was always fruitful; it helped me solving a number of critical problems. Rasul, Madhu helped me a lot whenever my computer troubled me. My heartfelt thanks goes to all of them.

I am also thankful to Mahesh, Girish, Yashesh, Shivani, Arpita, Soubhik, Abhishek, Mr. Ravinder Nath, Mr. A. Rajesh for their help during my stay over here. It will be very difficult for me to forget the happy, interesting moments with Adnan, Kshitij, Ram Bilas, Anup, Gopi, Jayanta, Shila, Rajib, Kausik, Rupen, Ashutosh, Pushpendra, Anurag, Brijesh, Rajesh, Dhananjay, Venkatesh, Rohit and other classmates, who made my IIT-life memorable.

Finally, I sincerely owe to my parents and the Great Guru, Shri Prabhat Ranjan Sarkar. Their continuous encouragement, support, blessing, guidance have brought me to the portal of my present achievement.

Kausik Maiti

Contents

1 **Introduction** 1

1.1 Does video compression require any further research ? 1

1.2 Video coding: present scenario 3

1.3 3D video coding: pros and cons 6

1.4 Motivation behind the work 7

1.5 Organization of thesis 8

2 **Very Low Bit Rate (VLBR) Video Coding** 9

2.1 Some observations 9

2.2 Proposed coding strategy 13

2.3 Structure of the proposed codec 14

2.4 Key operations and algorithms 18

2.4.1 Discrete cosine transform 18

2.4.2 Subband coding and wavelets 19

2.4.3 Set partitioning in hierarchical tree (SPIHT) 23

2.5 Important implementation issues 25

2.6 Simulation result 31

3 **VLBR Video Coding: An Improved Codec** 47

3.1 Problem identification 47

3.2 A possible solution 51

3.3 Implementation Issue 52

3.4 Simulation result 53

4 Video Coding : A Generalized Approach **62**

4.1 Generalization : why & how? 62

4.2 Design of codebook 64

4.2.1 Initial codebook design 65

4.2.2 Iterative codebook updating 68

4.3 General structure of the proposed codec 69

4.4 Simulation result 70

5 Scalable Coding: A Much Desired Feature **74**

5.1 Scalability: why is it needed? 74

5.2 Scalability: its different forms 77

5.2.1 Temporal Resolution Scalability 77

5.2.2 Spatial Resolution Scalability 77

5.2.3 Data-rate Scalability 78

5.3 Potential of the proposed method 78

6 Conclusion And Future Scope **83**

List of Figures

1.1 Growth of Internet Traffic Over Recent Years 2
1.2 Potential Application Areas of Different Video Coding Standards . . . 4
1.3 Block Diagram of Conventional Hybrid Codec 4
1.4 Demonstrating Parent-Child Relationship used in 3DSPIHT Algorithm 8

2.1 Miss America: A Typical Head & Shoulder Type Video 10
2.2 Side View: Temporal Changes Along Vertical Plane 'V' 10
2.3 Vertical View: Temporal Changes Along Horizontal Plane 'H' 11
2.4 Salesman: Another Typical Head & Shoulder Type Video 12
2.5 Side View: Temporal Changes Along Vertical Plane 12
2.6 Vertical View: Temporal Changes Along Horizontal Plane 13
2.7 Internal Block Diagram of the Proposed Video Coder 15
2.8 Internal Block Diagram of the Proposed Video Decoder 17
2.9 A K-Level One-Dimensional Wavelet Decomposition 19
2.10 A Typical Test Image 20
2.11 The Subband Labeling Scheme for an One-Level, 2D Wavelet Transform 21
2.12 One-Level 2D Wavelet Decomposition of the Image Shown in Figure 2.10 21
2.13 3-Level 2D Wavelet Decomposition of the Image Shown in Fig. 2.10 . . 22
2.14 Parent-Offspring Dependencies in the Spatial Orientation Tree 22
2.15 The Overall Bit-stream Structure 26
2.16 Example Histograms for Different Frequency Planes 29
2.17 Akiyo : Coded at 122 Kbps (Compression Ratio 200:1) 34
2.18 Akiyo : Coded at 122 Kbps (Compression Ratio 200:1) 35

2.19 Claire : Coded at 122 Kbps (Compression Ratio 200:1) 36
2.20 Claire : Coded at 122 Kbps (Compression Ratio 200:1) 37
2.21 Miss America : Coded at 122 Kbps (Compression Ratio 200:1) 38
2.22 Miss America : Coded at 122 Kbps (Compression Ratio 200:1) 39
2.23 Hall Monitor : Coded at 122 Kbps (Compression Ratio 200:1) 40
2.24 Hall Monitor : Coded at 122 Kbps (Compression Ratio 200:1) 41
2.25 Salesman : Coded at 152 Kbps (Compression Ratio 160:1) 42
2.26 Variations in Frame-PSNR for 'Akiyo' Sequence at 122Kbps 43
2.27 Variations in Frame-PSNR for 'Claire' Sequence at 122Kbps 43
2.28 Variations in Frame-PSNR for 'Miss America' Sequence at 122Kbps . . 44
2.29 Variations in Frame-PSNR for 'Hall Monitor' Sequence at 122Kbps . . 44
2.30 Variation in Frame-PSNR for "Salesman" Video Sequence at 152Kbps . 45
2.31 Performance Comparison at 0.04bpp : (a) Original 32^{rd} Frame of 'Claire', (b) Reconstructed Frame Using the Proposed Codec, (c) Reconstructed Frame Using 3DSPIHT 45
2.32 Performance Comparison at 0.04bpp : (a) Original 46^{th} Frame of 'Claire', (b) Reconstructed Frame Using the Proposed Codec, (c) Reconstructed Frame Using 3DSPIHT 46

3.1 First Four Frequency Planes Obtained After Temporal Decorrelation of the First GOF of 'Claire' 48
3.2 Two Levels of Wavelet Decomposition of the Third Frequency Plane . . 49
3.3 Frequently Used Scanning Patterns 50
3.4 (a) The Second AC Plane, (b) The Mask 51
3.5 32^{nd} Frame of 'Claire' : Coded at 60.8Kbps (Compression Ratio 400:1) 53
3.6 70^{th} Frame of 'Claire' : Coded at 60.8Kbps (Compression Ratio 400:1) 54
3.7 88^{th} Frame of 'Claire' : Coded at 60.8Kbps (Compression Ratio 400:1) 54
3.8 124^{th} Frame of 'Claire' : Coded at 60.8Kbps (Compression Ratio 400:1) 55
3.9 31^{st} Frame of 'Claire' : Coded at 122Kbps (Compression Ratio 200:1) . 55
3.10 62^{nd} Frame of 'Claire' : Coded at 122Kbps (Compression Ratio 200:1) . 56
3.11 84^{th} Frame of 'Claire' : Coded at 122Kbps (Compression Ratio 200:1) . 56

3.12 136th Frame of 'Claire' : Coded at 122Kbps (Compression Ratio 200:1) 57
3.13 66th Frame of 'Hall Monitor' : Coded at 60.8Kbps (Compression Ratio 400:1) 57
3.14 125th Frame of 'Hall Monitor' : Coded at 60.8Kbps (Compression Ratio 400:1) 58
3.15 183rd Frame of 'Hall Monitor' : Coded at 60.8Kbps (Compression Ratio 400:1) 58
3.16 267th Frame of 'Hall Monitor' : Coded at 60.8Kbps (Compression Ratio 400:1) 59
3.17 24th Frame of 'Hall Monitor' : Coded at 122Kbps (Compression Ratio 200:1) 59
3.18 57th Frame of 'Hall Monitor' : Coded at 122Kbps (Compression Ratio 200:1) 60
3.19 130th Frame of 'Hall Monitor' : Coded at 122Kbps (Compression Ratio 200:1) 60
3.20 214th Frame of 'Hall Monitor' : Coded at 122Kbps (Compression Ratio 200:1) 61

4.1 Side View: Temporal Changes Along Vertical Plane 63
4.2 Vertical View: Temporal Changes Along Horizontal Plane 63
4.3 $d_1 d_2$ number of d_3 dimensional data-points 67
4.4 Temporal Decorrelator in the Generalized Encoder 69
4.5 Performance Comparison Between 'codec 1' and 'codec 2' 71
4.6 Original 73rd Frame of 'Irene' Sequence 71
4.7 73rd Frame of 'Irene' : Coded at 304Kbps 72
4.8 73rd Frame of 'Irene' : Coded at 506Kbps 72
4.9 73rd Frame of 'Irene' : Coded at 1Mbps 73
4.10 73rd Frame of 'Irene' : Coded at 2Mbps 73

5.1 A Heterogeneous Communication Network with Video Services. 75
5.2 The Modified Bit-stream Structure 79

5.3 Subband Decomposition Required To Achieve The Modified Bit-Stream Structure 80

List of Tables

2.1 Psycho-Visual Weights For Different Frequency Planes 31

Chapter 1

Introduction

1.1 Does video compression require any further research ?

After the establishment of MPEG-2 in 1994, there were several voices stating that it was no more worth to put more effort into video coding. Because the then achieved reduction to 1...2 % of the original data rate was considered as sufficient. Even at the initial stage of standardization of MPEG-4, more emphasis was put on incorporating additional functionalities rather than on increasing the compression rate, though in the last phase, once again the issue of increasing compression ratio drew attention. And finally with the definition of the ACE (Advanced Compression Efficiency) Profile the coding efficiency could be increased by 30...50 % as compared to MPEG-2. Doubling the compression rate has also become the target of H.26L [16], the follow-up standard of H.263++ currently being developed within ITU-T SG16.

Today, when we already have huge amount of bandwidth available in our hand (as for example, today's glass fibers can carry terabits/sec, broadband Internet access via cable modems or xDSL is becoming widely available, today's DVDs and harddiscs can store hours of video), when every ten years the cost of digital storage is getting reduced by a factor of 100 or more (and this trend is expected to continue for coming next 20 years), a question naturally comes to our mind : why are we still continuing

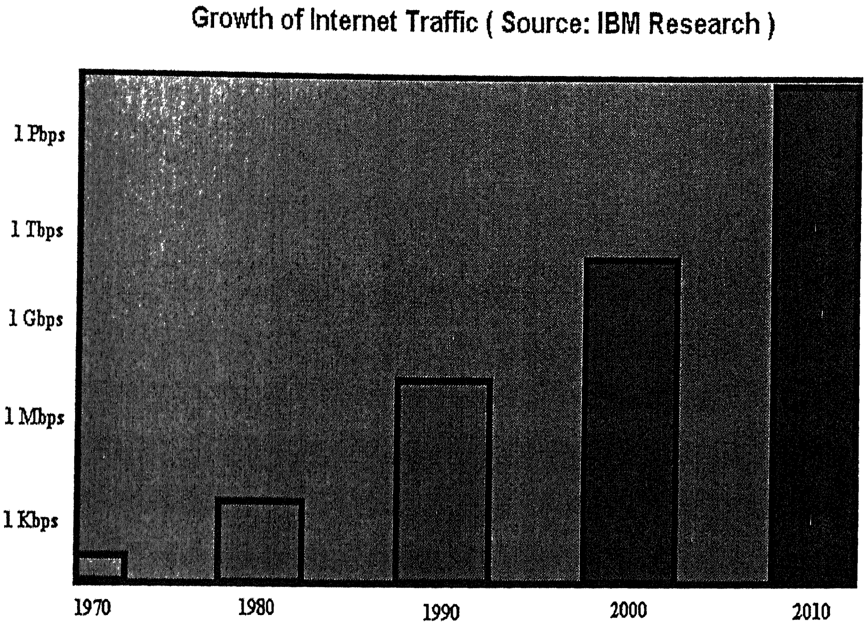


Figure 1.1: Growth of Internet Traffic Over Recent Years

R&D in video compression? The algorithms are becoming more and more complex and the improvements are becoming more and more marginal. Then, what is still missing today and what are the applications still asking for higher compression rates?

The answers to these questions can be debated among today's video coding research groups. However, we believe what is still missing is a universal full-featured video-compression technique that will perform equally well for wide range of video coding applications. And the application field that still demands higher compression rates is mobile video services. Spectrum over the air is a scarce resource which cannot be enlarged. Even with the advent of UMTS/IMT 2000, the available data rate will be restricted to 144 kbit/s for mobile reception, higher rates up to 2 Mbit/s will only be available in microcells, i.e. in restricted areas like offices. It's a bare fact that 144 kbit/s is still a rather limited data rate for video transmission, if available coding schemes like MPEG-4 or H.263++ are used. On the other hand, mobile Internet traffic will grow at the same speed as Internet traffic in general (as the number of mobile terminals is exploding day by day). It is therefore quite foreseeable that the air bandwidth will remain a permanent bottleneck, which justifies all efforts to squeeze video even further.

1.2 Video coding: present scenario

Since the beginning of the 20th century, video coding has evolved from mostly an academic R&D field into a highly commercial business. The demand for image sequence coding has increased in leaps and bounds over last few decades mainly due to the factors like increased availability of personal workstations, Internet, multimedia and also because of increased urge of information societies for better interaction among themselves. This in turn has motivated to build video coding standards. Standards have allowed easy interoperability across a wide range of equipments.

Since 1990, a series of video compression standards have been effected by two main standardization authorities : ISO/IEC and ITU-T. Each of these standards aims at different target application. As for example, MPEG-1 (standardized by ISO/IEC in the year 1991) is made for coding video at around 1.5Mbps. Potential application area is Video-CD. MPEG-2 (standardized by ISO/IEC in the year 1994) is optimized for HDTV. Target bit rate is 2-60 Mbps. Around 4 Mbps the quality of image achieved is similar to PAL/NTSC/SECAM. MPEG-4 (standardized by ISO/IEC in the year 1999) is basically for multimedia application. It can code video at very low bit rate e.g. around 64 Kbps. And finally H.26x, video compression standards by ITU-T are focussed to low bit rate video coding applications.

However different may these standards appear, all of them share a common coding strategy. They all use hybrid coding scheme (motion estimation and motion compensation followed by discrete cosine transform (DCT)). Now, as far as the coding performance is concerned, this hybrid scheme seems quite satisfactory. Yet, it is not very perfect. It suffers from some serious drawbacks like high computational complexity, intense blocking artifacts and mosquito noise at low bit rates, possibility of infinite error-propagation. And last but not the least, it provides inefficient support for scalable representation and coding, the most demanded feature of today's video coding.

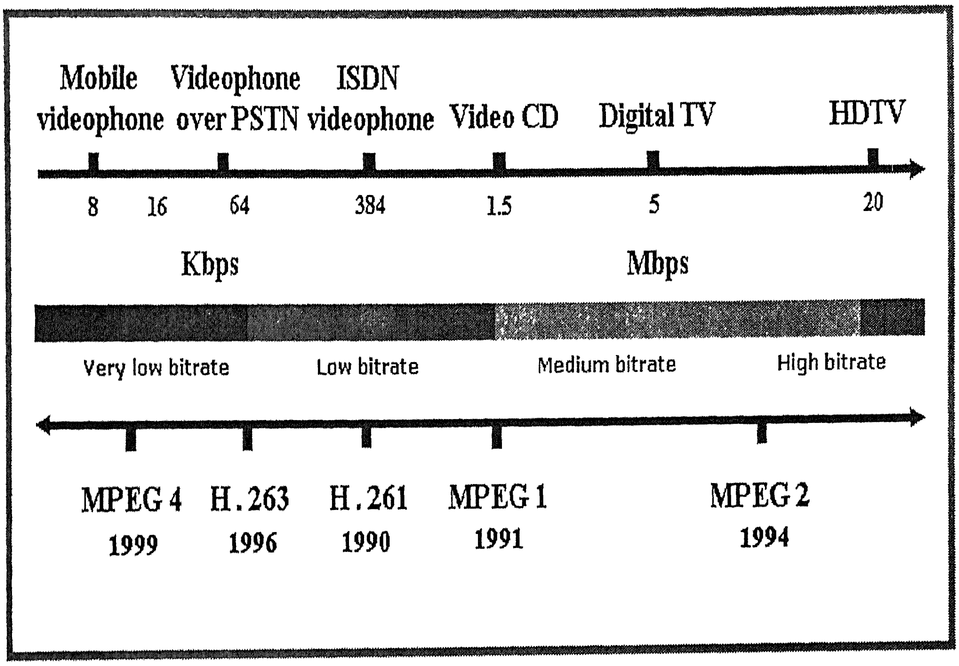


Figure 1.2: Potential Application Areas of Different Video Coding Standards

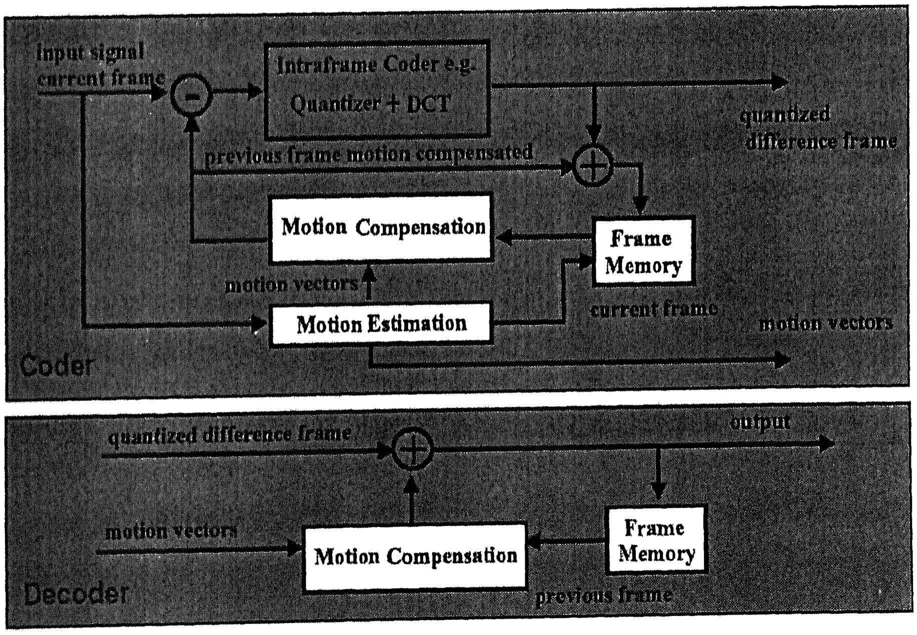


Figure 1.3: Block Diagram of Conventional Hybrid Codec

Researchers have naturally been motivated to find out some alternative approaches. As a result, lots of non-standard techniques have been proposed in the last few years. Three dimensional (3D) transform coding is the most remarkable among them. (In this thesis, we shall confine ourselves mainly into this particular mode of video compression). As a part of "3D video coding" strategic movement, we have seen people making use of 3D DCT so as to code group of consecutive video frames (GOF) [1], though, it did not ultimately gain much appreciation mainly because of its heavy computational complexity. Also the blocking artifact could not be eliminated due to independent block-by-block processing of video. However, 3D subband coding (SBC) using wavelets gave a major break-through with a number of inherent attractive features and has emerged as the most viable alternative of today's video coding standards.

At the early stage, 3D subband coding schemes had been designed and applied for mainly high or medium bit-rate video coding. Karlsson and Vetterli [2] took the first step toward 3D-SBC using a simple 2-tap Haar filter for temporal filtering. Podilchuk, Jayant, and Farvardin [3] used the same 3D subband framework without any motion compensation. It employed adaptive differential pulse code modulation (DPCM) and vector quantization to overcome the lack of motion compensation. Kronander [4] first presented motion compensated temporal filtering within the 3D SBC framework. Ohm [5] and later Choi and Woods [6] refined the method and utilized different quantization techniques in application to subbands produced by perfect reconstruction filter banks.

In a parallel effort, in order to capture the multiresolutional nature of SBC schemes and to generate a scalable (both fidelity wise and resolution wise) bit stream, several scientists came out with their innovative ideas. Bove and Lippman [7] first proposed multiresolutional video coding with a 3D subband structure. Taubman and Zakhor [8] introduced a multi-rate video coding system using global motion compensation for camera panning, in which the video sequence was predistorted by translating consecutive frames before temporal filtering with 2-tap Haar filters.

However, it was only after 1993, when J. M. Shapiro [12] introduced embedded zerotree wavelet (EZW) coder for still image coding, particularly when Pearlman and Said [10] launched their SPIHT (Set Partitioning In Hierarchical Trees) codec (one of the most successful improved EZW coders) in 1996, research on wavelet based visual information coding gained actual momentum. Lots of attempts have been made thereafter to extend the idea behind those two coders to exploit the temporal redundancy present in video signal. Chen and Pearlman [13] proposed 3D improved embedded zerotree wavelet (IEZW) coder for video and showed promise of an effective and computationally simple video coding system without motion compensation. Another highly scalable embedded 3D SBC system [14] with tri-zerotrees for low bit-rate environment was reported with coding results visually comparable, but numerically slightly inferior to H.263. Following the same line, in 2000, Kim and Pearlman proposed 3D extension of SPIHT coder for low bit rate application [9]. They applied 3D dyadic wavelet to every GOF and coded the wavelet coefficients by 3D SPIHT. Even without any motion estimation and compensation this method performs measurably and visually better than MPEG-2. Finally, in 2002, a full-featured, error-resilient, scalable version of 3D SPIHT algorithm has been suggested by Sungdae Cho and William A. Pearlman [15].

Today, the way research on 3D wavelet based video coding is progressing, its seems apparent that in near future, it will give birth to some new video coding standard with 3D wavelet coding as its kernel.

1.3 3D video coding: pros and cons

3D wavelet video coding was proposed with the notion to extend the wavelet decomposition so as to include the time domain as well. The reported advantages of 3-D wavelet video coding include (1) symmetricity, (2) low computational complexity at both encoder and decoder end, (2) efficient scalability support, multiresolution transmission and display, (3) robustness of the generated bit-stream against transmission loss (due to absence of both spatial prediction as well as recursive loop in the codec architecture) and (4) absence of blocking artifacts, possibility of the use of psycho-visual weights in order to adjust the coding to the sensibility of the human optic sense.

Among the disadvantages, large storage space requirement at both encoder and decoder side, longer processing delay are the main two points. Also, due to GOF-by-GOF processing, since the correlation between boundary frames can not be taken into account, PSNR drops by several dB at the GOF boundaries. This gives rise to temporal jerk while playing back the decoded video.

1.4 Motivation behind the work

Kim and Pearlman's 3D SPIHT coder [9] was able to draw appreciation only because it achieved good objective as well as subjective quality (comparable with H.263) with minimal computational complexity. Noteworthy, no motion compensation was used there for coding low-motion-video. 3D dyadic wavelet was applied to decompose a GOF into several spatio-temporal subbands. 3D SPIHT was then used to quantize and code the wavelet coefficients. But, the point that is of our interest is, 3D SPIHT requires that the level of decomposition along two spatial dimensions and the time dimension must be same. Also, at the coarsest subband, there must be atleast 8 coefficients (refer to Figure 1.4). This limits the flexibility of the codec. Supposing that the number of video frames in every GOF be 8, in order to satisfy those requirements the maximum allowable level of decomposition becomes 2. However, this 2 level of decomposition can not fully exploit the inter-pixel redundancy present in any frame (let it have any standard size, be it QCIF or CIF). Thus, we feel that there is still scope for improvement as far as compression performance is concerned.

To cope up with the situation when there is considerable amount of motion either in the form of global camera motion or local object motion within a GOF, a motion compensation scheme was kept as an option. Now, needless to say, motion estimation itself is a computationally intensive task. Furthermore, the motion vectors (to be sent to the decoder end) simply increase the overhead information bits, which may count significant especially at very low bit rates.

With these two points in mind, we have made an attempt to build a coding architecture that will be capable of performing satisfactorily (at least visually) at very low bit rates (also at higher bit rates) without using motion compensation at all. We maintain, it is the subjective quality and not the objective one that matters most. Be-

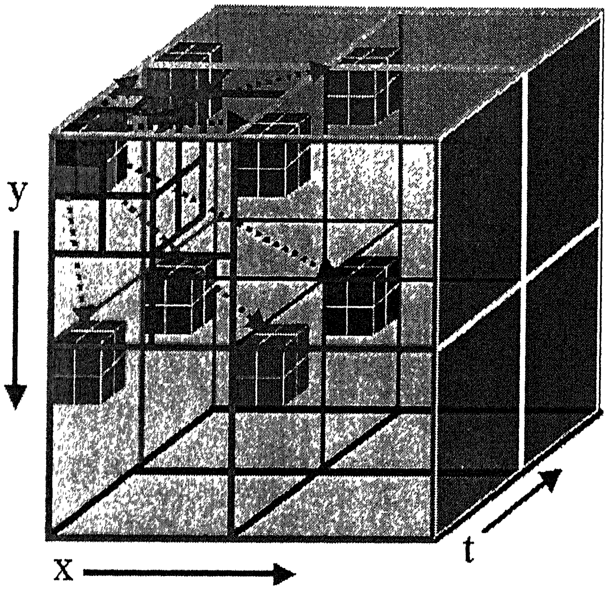


Figure 1.4: Demonstrating Parent-Child Relationship used in 3DSPIHT Algorithm

cause, after all, the decoded video quality will be judged by some human being whose satisfaction depends solely on subjective aspects of the video.

1.5 Organization of thesis

The entire thesis is organized as follows. Chapter 2 describes the proposed video coding algorithm that is suitable for coding low-motion-video at very low bit rate. Chapter 3 describes a simple modification on the previous coder so as to assure good visual quality at extreme low bit rate. In chapter 4, we present a generalized version of the proposed codec that is capable of coding video (natural as well as synthetic video with complex object motion) at wide range of bit-rates. Next in chapter 5, we have discussed how our coding scheme can include interesting features like fidelity-scalability, resolution-scalability and produce embedded bit-stream. Finally, we conclude and intimate the scope for future work in chapter 6.

Chapter 2

Very Low Bit Rate (VLBR) Video Coding

Very low bit rate video coding is particularly important for video-conferencing or video-telephony type applications. In this chapter, we suggest a simple coding algorithm and show that it performs quite satisfactorily at very low bit rate (0.04bpp) without even requiring any motion-compensation.

2.1 Some observations

Before going into detail of our codec structure, let us view at different videos (that can be coded at very low bit rates) from slightly different angle. Figure 2.1 shows a collection of consecutive frames of a head & shoulder type video: "Miss America". This typical test video is widely used to check the performance of very low bit rate video coders. It possesses very good intra-frame as well as inter-frame inter-pixel correlation. To examine exactly how the temporal changes occur, we have bisected the entire collection of frames along two planes, each being perpendicular to the video frames. Positions of these two planes have been adjusted so as to capture the regions containing larger object movements. Figure 2.2(a) shows the side view and Figure 2.3(a) presents the vertical view for collection of 150 frames. Figure 2.2(b) and Figure 2.3(b)

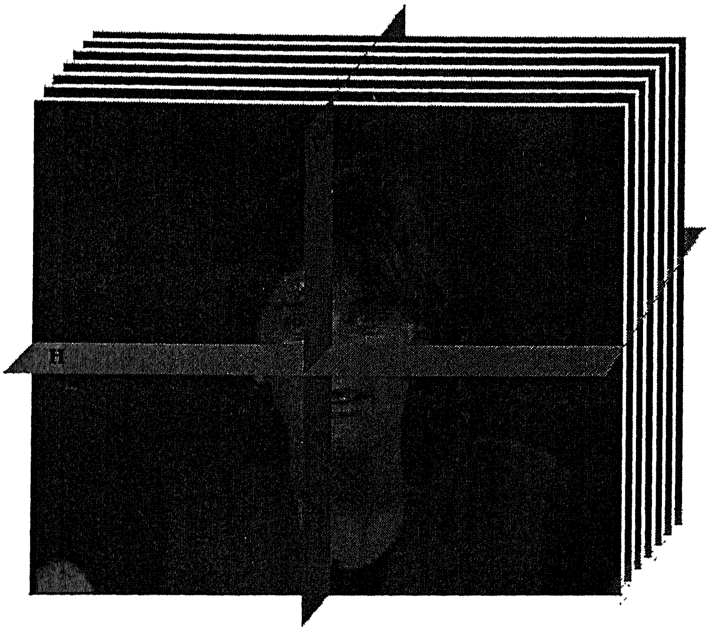
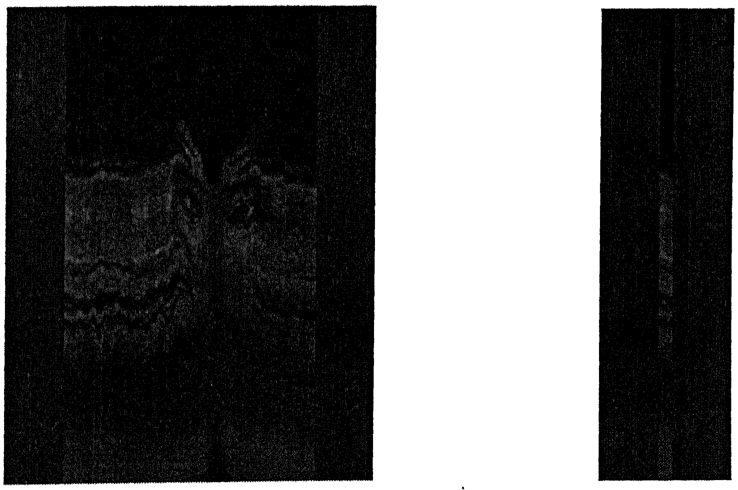


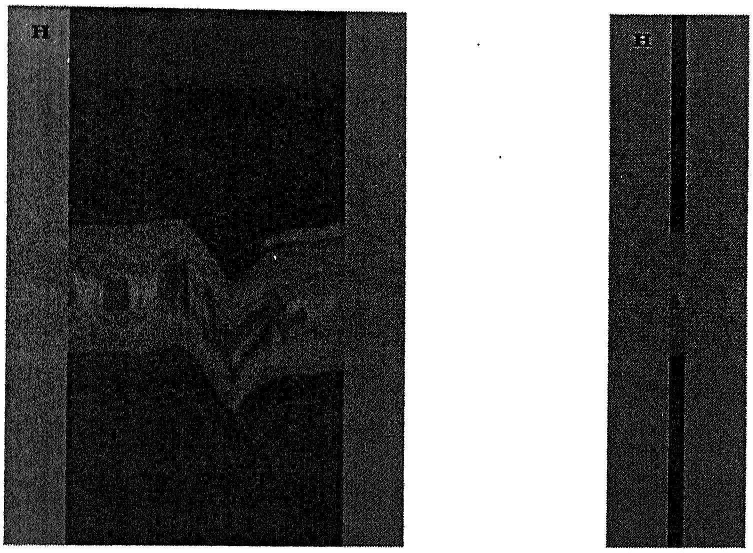
Figure 2.1: Miss America: A Typical Head & Shoulder Type Video



Number of frames: 150
(a)

Number of frames: 8
(b)

Figure 2.2: Side View: Temporal Changes Along Vertical Plane 'V'



Number of frames: 150
(a)

Number of frames: 8
(b)

Figure 2.3: Vertical View: Temporal Changes Along Horizontal Plane 'H'

display the side view and the vertical view respectively for collection of 8 frames (usual number of frames in GOF). Both of these figures clearly indicate the huge amount of temporal redundancy present among successive frames. Indeed, in most of the regions, except at the moving object-edges, there is no significant changes as such.

Figure 2.4 shows a group of successive frames of another low-motion video: "Salesman". This particular video, unlike "Miss America" sequence, contains large number of static as well as moving edges in each frame. However, as it can be seen from Figures 2.5 and 2.6, the amount of temporal redundancy is still enormous. Same thing can be observed in case of any "head & shoulder" type video or any low-motion-video that is suitable for very low bit rate coding. There may be varieties of textures, curves, edges etc. in the video-frames, but, the change of those patterns with time is, in fact, significantly less.

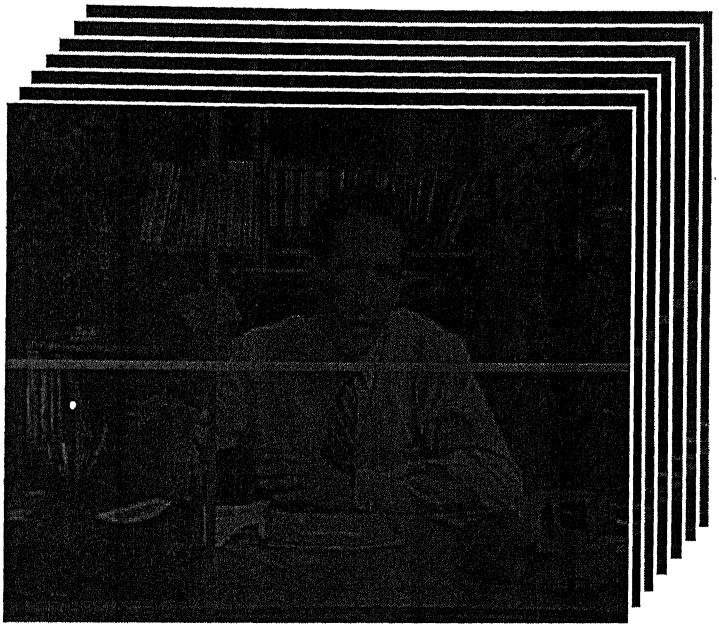


Figure 2.4: Salesman: Another Typical Head & Shoulder Type Video

Number of frames: 128
(a)

Number of frames: 8
(b)

Figure 2.5: Side View: Temporal Changes Along Vertical Plane

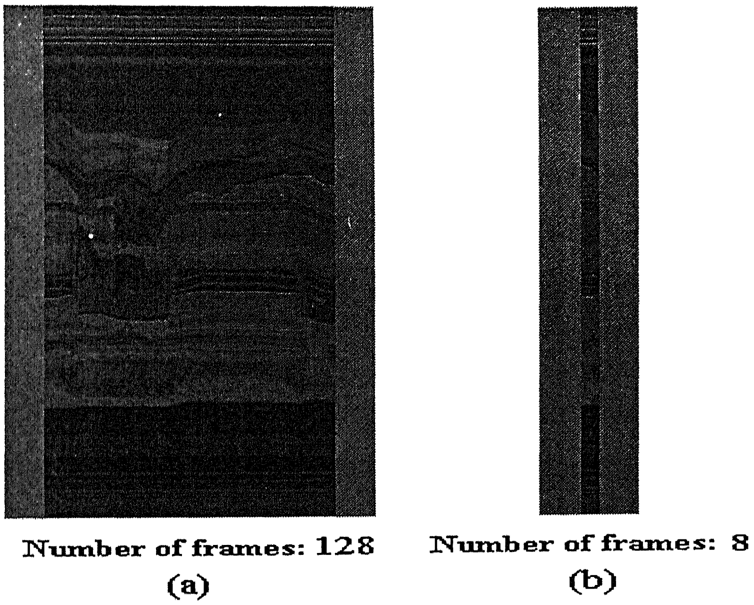


Figure 2.6: Vertical View: Temporal Changes Along Horizontal Plane

2.2 Proposed coding strategy

In many of the recent papers on 3D subband video coding, subband filtering has been applied on the GOF both along the spatial dimensions and along the temporal dimension. The aim is to capture video-information in different spatio-temporal subbands. Application of subband coding to individual video-frames can be well justified with our experience of wavelet based still image coding. However, use of the same along time axis can not be well justified. We believe, the advantage of temporal filtering can actually be realized if only if the entire video is taken at a time as a consolidated information block, which is simply not feasible.

We have adopted the strategy that may seem to lie somewhere midway between 3D SBC & 3D DCT. The idea is to combine strengths of both DCT and SBC so as to improve the performance of our codec. We have used DCT (Discrete Cosine Transform) to exploit the temporal redundancy present in the video signal. Needless to say, DCT has excellent energy compaction property for highly correlated data [19]. Also, DCT can play the role of motion compensation in more efficient way. In fact, it reduces the inter-frame redundancy among larger number of consecutive frames [1]. (The actual number of frames depends on the number of frames in a GOF). To exploit the spatial redundancy, however, we have retained subband (wavelet) decomposition technique. The motivation has been obtained partly from the success story of wavelet for still image compression. Also, efficient high performance wavelet coders (e.g. SPIHT, SLCCA[29], EBCOT[30]) can be made use of. In particular, among many of the inherent attractive features of subband coding [20], we are basically interested in the following few.

- Since the subband (wavelet) filters operate on the entire frame, the blocking effect can be reduced to great extent.
- Subband coding provides us scalability (both fidelity wise and resolution wise) as a free gift.
- Subband coding allows better bit rate control during encoding / decoding process.
- Subband coding is more robust under transmission or decoding errors. Because, errors on a particular subband may be masked by the information on the other subbands.

2.3 Structure of the proposed codec

The structure of the encoder has been shown in Figure 2.7 in form of block diagram. It consists of five important blocks : (1) video buffer, (2) temporal decorrelator, (3) spatial decorrelator, (4) bit-allocator and (5) quantizer & coder. For the sake of simplification, bit-allocator has not been explicitly shown in the diagram.

Our coder accepts a pre-processed video. By pre-processing we mean noise reduction and/or temporal sub-sampling (sometimes, spatial sub-sampling, too) of a raw video-data. This pre-processed video is buffered, first. The length of the buffer is $2L$, where L is the number of frames in a GOF. The reason behind choosing a buffer of double length is to allow video buffering and video encoding to continue simultaneously

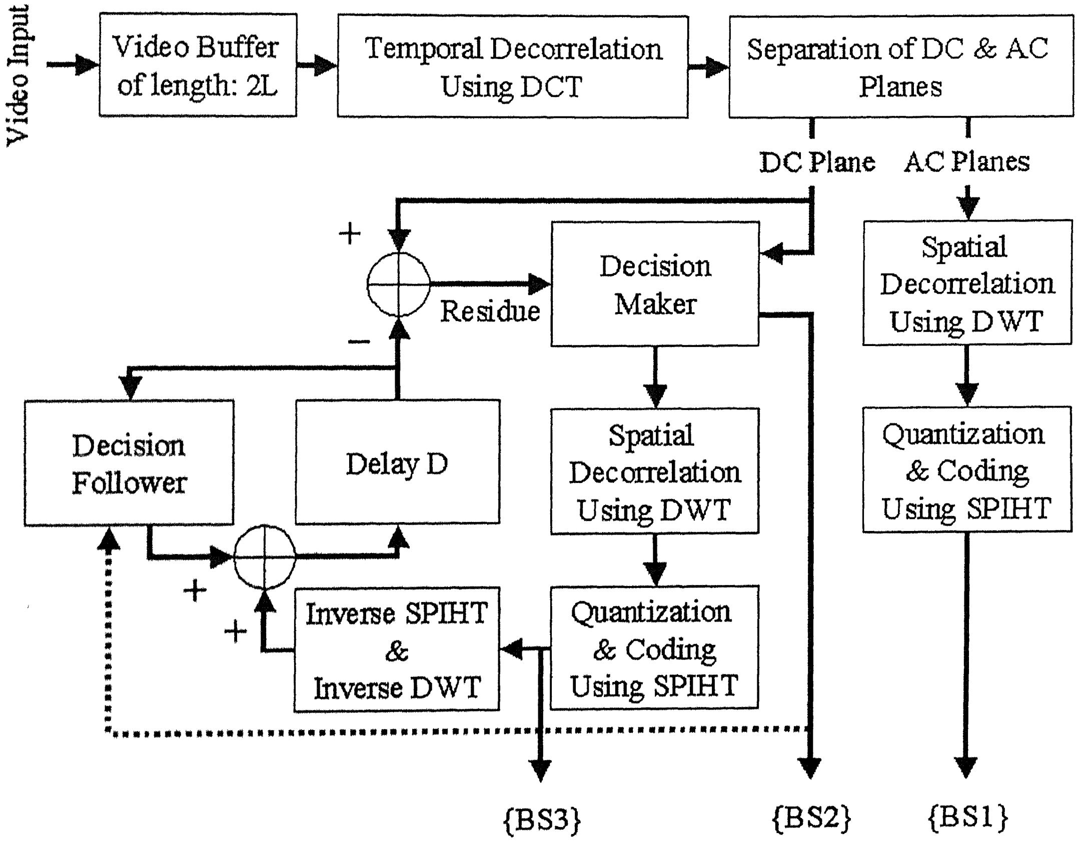

Video Input
Video Buffer of length: 2L
Temporal Decorrelation Using DCT
Separation of DC & AC Planes
DC Plane
AC Planes
+
−
Residue
Decision Maker
Spatial Decorrelation Using DWT
Decision Follower
Delay D
Spatial Decorrelation Using DWT
Quantization & Coding Using SPIHT
+
+
Inverse SPIHT & Inverse DWT
Quantization & Coding Using SPIHT
{BS3}
{BS2}
{BS1}

Figure 2.7: Internal Block Diagram of the Proposed Video Coder

and thereby to avoid unnecessary delay. DCT is then applied on the GOF along temporal dimension. This helps to decorrelate data along time. DCT, indeed, compresses the energy toward low frequency side and reduces the amplitudes of higher frequency components. Now as, there are L number of frames in a GOF, applying DCT along the time dimension will give rise to L number of different frequency planes. Let us call the first (lowest) frequency plane as DC plane and others as AC planes. Each of the AC planes is then split into M $(= 3N + 1)$ number of subbands using N level of dyadic wavelet decomposition (spatial decorrelation). However, in case of DC plane, we adopt slightly different approach. We try to predict the DC plane obtained from current GOF with our knowledge about the DC plane of previous GOF. No motion compensation is used. We simply generate back the previous DC plane from the corresponding bit-stream using inverse process and straightway subtract it from the current DC plane. Either the residue thus obtained or the current DC plane itself can be used in subsequent processing (wavelet decomposition: spatial decorrelation). Decision regarding which one to be used is made by the Decision Maker. It calculates the energy of both the residue and the DC plane. If the energy of the residue is greater than some pre-determined percentage of the energy of the DC plane, in other words, when the prediction error is significant, the DC plane is used. Otherwise, the residue is used. Decision made by the Decision Maker is used by the Decision Follower to appropriately reproduce the previous DC plane. Finally, the wavelet coefficients corresponding to each planes are quantized and coded using SPIHT algorithm [10]. An important issue that plays a vital role in determining the ultimate coding performance is bit allocation. Given total bit budget per GOF, how to distribute that amount among different frequency planes? To solve this problem, we have followed the approach given in [17] with little modification. The actual method has been discussed in section 2.5. Finally, our encoder generates three bit streams: {BS1}, {BS2} and {BS3}. {BS1} contains information regarding the AC planes, {BS2} contains the decision information and {BS3} carries information about the DC-plane/residue.

As a part of decoding process (refer to Figure 2.8), we simply carry out the reverse procedure.

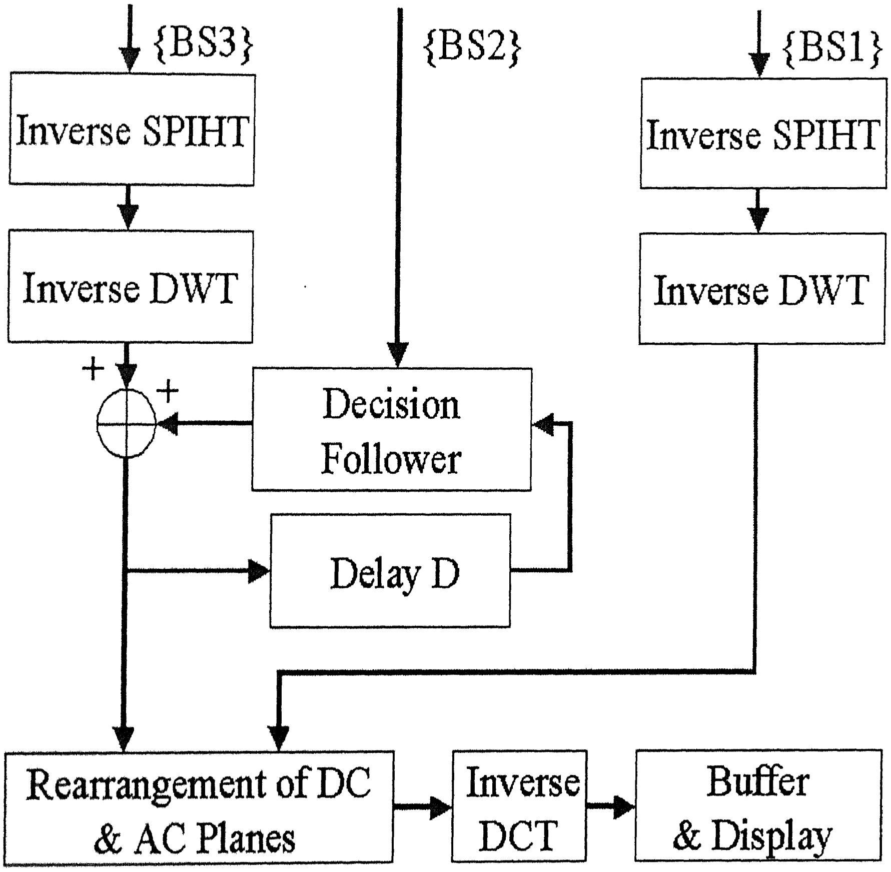


Figure 2.8: Internal Block Diagram of the Proposed Video Decoder

2.4 Key operations and algorithms

In this section, we briefly give some preliminary ideas about the key operations and algorithms involved in our codec.

2.4.1 Discrete cosine transform

Discrete cosine transform (DCT) is a fast linear transform that has been widely used in the past in different image/video compression applications. It has also become a part of most of the previous image/video coding standards. The one-dimensional DCT of a sequence $\{\ u(n),\ 0 \leq n \leq N-1\ \}$ is defined as [19]

$$v(k) \;=\; \alpha(k)\sum_{n=0}^{N-1} u(n)\cos\left[\frac{\pi(2n+1)k}{2N}\right], \qquad 0 \leq k \leq N-1 \tag{2.1}$$

where

$$\alpha(0) \;=\; \sqrt{\frac{1}{N}}, \quad \alpha(k) \;=\; \sqrt{\frac{2}{N}} \qquad for \qquad 1 \leq k \leq N-1 \tag{2.2}$$

The inverse transformation is given by

$$u(n) \;=\; \sum_{k=0}^{N-1} \alpha(k)\ v(k)\ \cos\left[\frac{\pi(2n+1)k}{2N}\right], \qquad 0 \leq n \leq N-1 \tag{2.3}$$

If we denote $\{\ u(n),\ 0 \leq n \leq N-1\ \}$ by U and $\{\ v(k),\ 0 \leq k \leq N-1\ \}$ by V, we can also put Equation (2.1) and Equation (2.3) as follows:

$$V = CU$$

$$U = C^T V$$

where $C \;=\; \{c(k,n)\}$, C^T is the transpose of C and

$$c(k,n) \;=\; \begin{cases} \sqrt{\frac{1}{N}}, & k=0, \quad 0 \leq n \leq N-1 \\ \sqrt{\frac{2}{N}}\ \cos\frac{\pi(2n+1)k}{2N}, & 1 \leq k \leq N-1, \quad 0 \leq n \leq N-1 \end{cases} \tag{2.4}$$

As a matter of fact, DCT of a vector having length N can be computed in $O(N\log_2 N)$ operations via an N-point FFT [19].

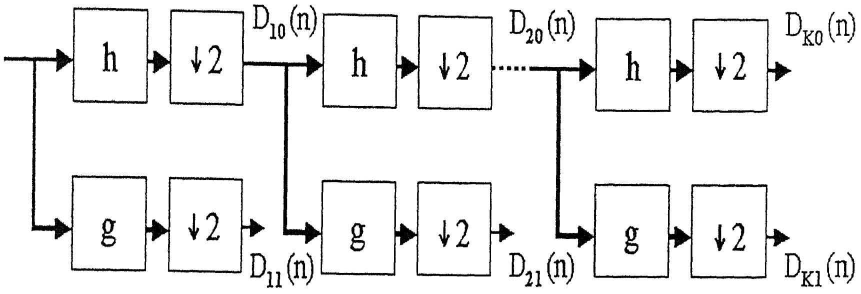


Figure 2.9: A K-Level One-Dimensional Wavelet Decomposition

2.4.2 Subband coding and wavelets

Subband coding is an important imaging technique with ties to multiresolution analysis [18]. Wavelet coding also comes under the same category. In both cases, the input signal is decomposed into multiple subbands using combination of low-pass and high-pass filters. These subbands are then subsampled, quantized and coded. At the decoder end, exactly inverse operations (upsampling and reassembling the subbands) are carried out to form an approximation to the input signal.

The main difference between subband and wavelet coding lies in the choice of the filters to be used in the transform. The filters used in wavelet coding systems are typically designed to satisfy certain smoothness constraints [22]. Whereas, the subband filters are designed mainly to satisfy the criteria of nonoverlapping frequency responses so as to achieve decorrelation among output subbands.

The generic form of a one-dimensional (1D) wavelet transform is shown in Figure 2.9. Here a signal is passed through a lowpass and a highpass filter, h and g, respectively, then downsampled by a factor of two, constituting one level of transform. Multiple levels or "scales" of the wavelet transform are obtained by repeating the filtering and decimation process on the lowpass branch outputs only. The process is typically carried out for a finite number of levels K. The resulting coefficients, $D_{11}(n)$, $D_{21}(n)$, ... $D_{K1}(n)$ *and* $D_{K0}(n)$ are called wavelet coefficients. Noteworthy, here, the downsampling factor is equal to the number of filters at each level. This par-

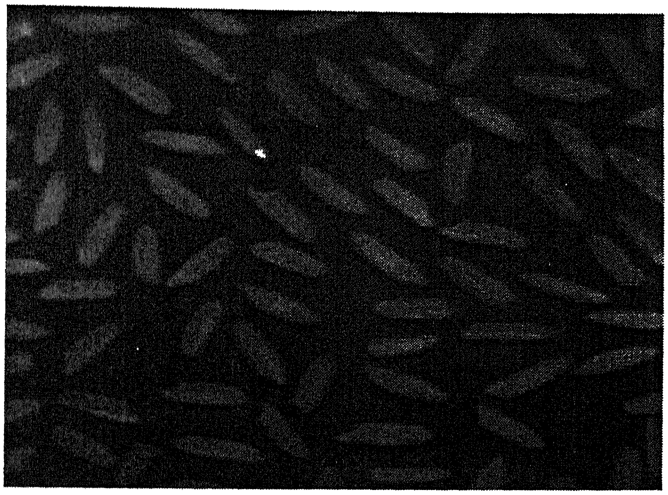


Figure 2.10: A Typical Test Image

ticular type of wavelet transform is termed as "maximally decimated form of wavelet transform".

The 1D wavelet transform can be extended to a two-dimensional (2D) wavelet transform using separable wavelet filters. With separable filters the 2D transform can be computed by applying a 1D transform to all the rows of the input, and then repeating on all of the columns. Figure 2.12 shows an example of a one-level ($K = 1$), 2D wavelet transform of the image shown in Figure 2.10, with corresponding notations given in Figure 2.11. This can further be repeated for any level of wavelet expansion (refer to Figure 2.13 for a 3-level wavelet decomposition) and so on.

Wavelet transform tends to compact the energy of the input into a relatively small number of wavelet coefficients. For example, in naturally occurring images, much of the energy in the wavelet transform becomes concentrated into the LL_K subband. In addition, the energy in the high frequency bands (HL_i, LH_i, HH_i) is also concentrated into a relatively small number of coefficients. Initially, designers of the wavelet coding systems recognized that low bit rate, low mean squared error (MSE) wavelet coders can be achieved by coding only few select high energy coefficients. However, the problem was how to code both the position information as well as the magnitude information for each of these coefficients so that data can be recovered properly at the decoder

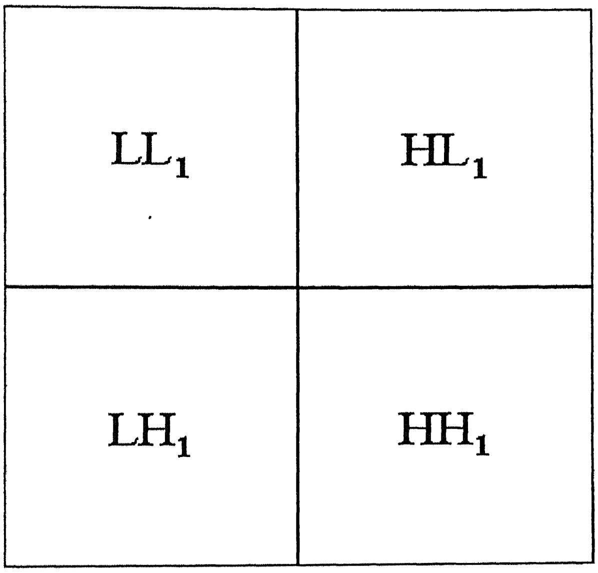


Figure 2.11: The Subband Labeling Scheme for an One-Level, 2D Wavelet Transform

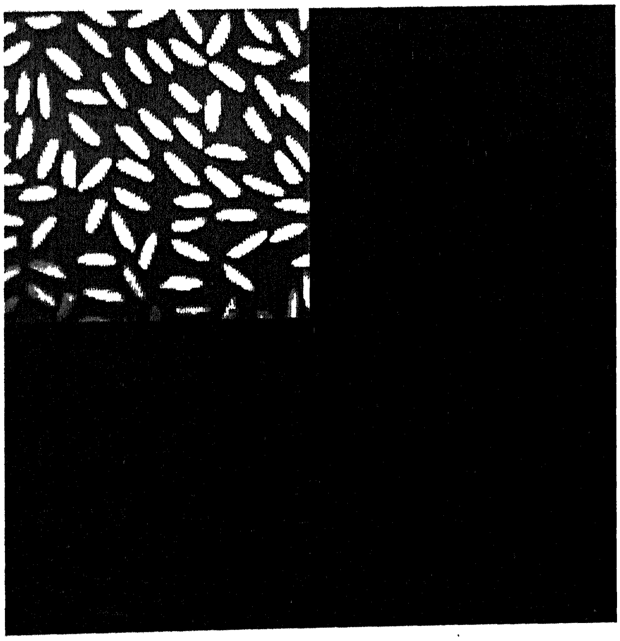


Figure 2.12: One-Level 2D Wavelet Decomposition of the Image Shown in Figure 2.10

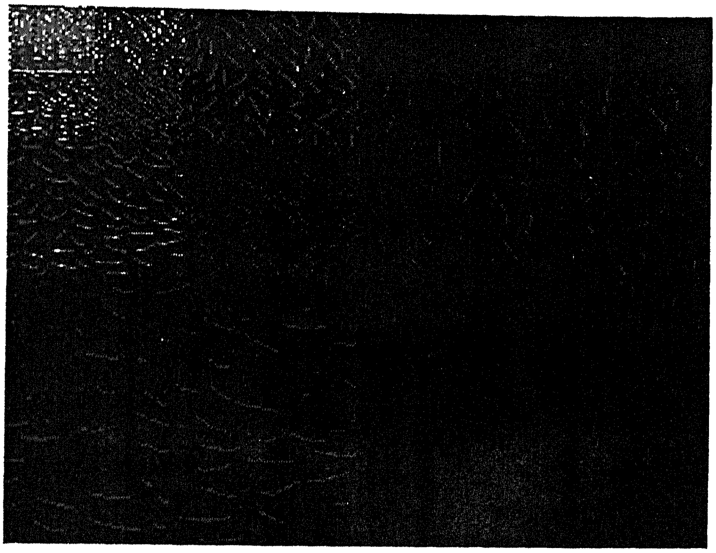


Figure 2.13: 3-Level 2D Wavelet Decomposition of the Image Shown in Fig. 2.10

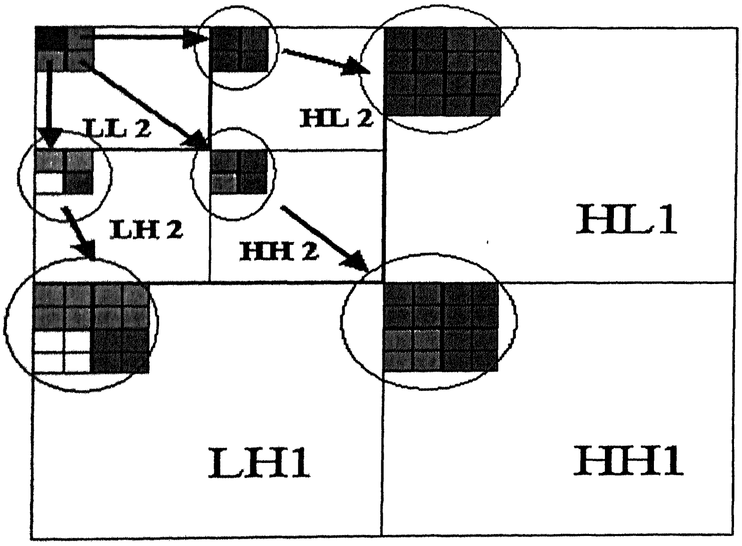


Figure 2.14: Parent-Offspring Dependencies in the Spatial Orientation Tree

end. In fact, depending on the method used earlier, the amount of resources required to code the position information sometimes turned out to be a significant fraction of the total, thereby negating much of the benefit of the energy compaction.

J. M. Shapiro [12] was the first person to show an efficient way to tackle this problem. In his pioneering work, he proposed a coder that simplifies the coding of the position informations of the significant wavelet coefficients using inter-band prediction (or prediction across frequency subbands) technique. The basic idea is to use the location of significant coefficients in one frequency band to predict the location and magnitude of significant coefficients in other frequency bands, thus reducing the cost of coding position information. Indeed, many of the modern wavelet coders (e.g. SPIHT [10]) too use almost the same idea. Inter-band prediction technique that exploits the self-similar, hierarchal nature of the wavelet transform, can easily be explained using Figures 2.12 and 2.13. Careful observation of these figures reveals that the significant coefficients in the high frequency subband do not occur at random locations, rather tend to cluster. Furthermore, these clusters tend to occur at the same relative spatial locations in each of the high frequency subbands which often correspond to discontinuities or edges that occur in the original image. This particular self-similar relationship has been defined using "spatial orientation tree" structure in [10].

2.4.3 Set partitioning in hierarchical tree (SPIHT)

In this section, we briefly outline the concepts and operations involved in SPIHT coding. Interested reader is requested to please refer to [10] for a more elaborate discussion.

As we have already mentioned that SPIHT coding utilizes the inter-band prediction technique that exploits the self-similar, hierarchal nature of the wavelet transform. Hence, at first, SPIHT algorithm defines a special tree structure called "spatial orientation trees" (SOT) to perfectly represent the self-similarity. Figure 2.14 shows how the "spatial orientation tree"s are defined in case of a two-dimensional dyadic wavelet transform with two levels of decomposition. Each node of the tree corresponds to a wavelet coefficient and is identified by its coordinate. Its direct descendants (offspring) correspond to the coefficients of the same spatial orientation in the next finer level of the pyramid. The tree is defined in such a way that each node has either no offspring or four offspring, which always form a group of 2 × 2 adjacent wavelet coefficients. The coefficients in the highest level of the pyramid are tree roots and are also grouped in 2 × 2 adjacent wavelet coefficients. However, their offspring branching rule is different, and in each group, one (e.g. one as indicated by a small black square in Figure 2.14) does not have any descendants. As the wavelet coefficients belonging to a SOT correspond to the same spatial location of original frame/image, it is very likely that if the magnitude of a coefficient in a certain node of a SOT does not exceed a given threshold value, then none of its descendants will exceed that threshold.

While encoding, SPIHT algorithm performs multiple scanning through the entire wavelet coefficient set. In each pass, it checks the magnitude of the coefficients against some threshold value. The initial threshold is computed as

$$T = 2^n \; where \; n = \left\lfloor \log_2 \left(\max_{(i,j)} \{ |c_{i,j}| \} \right) \right\rfloor$$

This threshold value is reduced to its half everytime a pass is completed. The scanning order is always kept fixed. The members of the "list of insignificant pixels" (LIP) are scanned first. Then the members of the "list of insignificant sets of pixels" (LIS) and finally the elements in the "list of significant pixels" (LSP) are scanned. Initially, LIP consists of the coordinates of all the wavelet coefficients in the coarsest subband, LIS consists those of all SOT-roots and LSP is kept empty. While scanning the coefficients pertaining to LIP, if anyone is found significant (i.e with magnitude greater than or equal to the threshold value), a "1" bit and the associated sign bit are emitted. The corresponding coordinate is moved from LIP to LSP. Otherwise, a "0" bit is sent. Next, while testing any set belonging to LIS, a "0" or a "1" bit is sent conveying the message whether all the elements in that tree are insignificant or not. If found not insignificant, the immediate four descendants of the current root of the tree are tested for significance. Any descendant found significant is moved to LSP with simultaneous emission of a "1" bit and sign bit. And the insignificant descendant is placed in LIP and a "0" bit is sent to the decoder. Once these four descendants are moved in this manner, the tree is treated as a "L"-type set. Previously, it used to be treated as "D"-type set. The "L" set is left aside for scanning second time in the same pass. It is to be scanned at the end. If however it turns out to be an empty set, it is completely eliminated from the scan-list. During scanning of the "L" set, as before, a "0" or "1" bit is emitted from the encoder to inform the result of significance-test to the decoder. In case the set is found to contain some significant element, it is split into four "D"-type trees. The current root of this newly formed "D"-type set is one of the four immediate descendants of the previous root element. Scanning of LIP and LIS elements in this way is termed as "sorting". In the "refinement" phase, the last phase of one pass, the n^{th} most significant bits of the coefficients that were entered into LSP in previous pass are sent as refinement bits. This way the coding process continues until the desired bit rate is reached.

In the decoder side, the same lists, LIP, LIS and LSP, are prepared and updated in each pass. Also the same scanning pattern and decision/branching rule is used. As the input bits are read from the codestream, the decoder reconstructs the magnitude and the sign bits of LSP members seen by the encoder. The coefficients of the final LIP and LIS sets are set to zero. Especially, in the wavelet transform of an image or a frequency plane, large sets of zero values exist which are identified efficiently by SPIHT with a single bit. Moreover, the significant coefficients are never represented by more bits than actually needed in their natural binary representation, since the highest "1" bit can always be known apriori in this process.

2.5 Important implementation issues

We have implemented the proposed codec in Matlab (Version 6.1.0.450, Release 12.1). No attention has been paid in optimizing our code as such. Therefore, at present, we do not make any statement whether our codec can be used in real time environment or not. But, we are very much hopeful about it. In the following, we try to highlight on some implementation issues that we feel, greatly influence the performance of our coder.

The first issue is the choice of wavelet filter. We have tried with different orthogonal, compactly supported wavelet filters as well as bi-orthogonal, compactly sup-

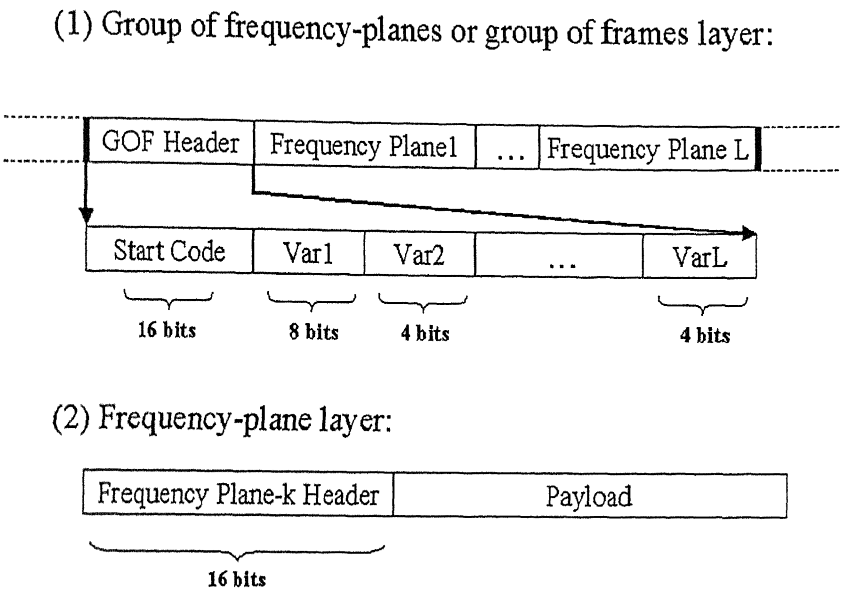


Figure 2.15: The Overall Bit-stream Structure

ported wavelet filter pairs available in Matlab. Finally, we have chosen Daubechies 9/7 bi-orthogonal wavelet filter pairs (both for analysis and synthesis purpose) because of its superior performance. By the term "superior", we mean that with the same wavelet quantizer, coder and decoder, among all other wavelet filters, Daubechies 9/7 bi-orthogonal wavelet filter pairs yield better quality (higher PSNR) of the decoded image.

The second issue is the amount of overhead bits required while coding. Especially at very low bit rate coding scenario, overhead information may consume significant part of the total bit budget and thus reduce the efficiency of the codec to great extent. Hence, this issue of overhead bits needs some attention too. In Figure 2.15, we show the overall bitstream structure and the overhead bits required for the proposed coding scheme. The bitstream, in fact, is arranged in a hierarchical structure composed of the following two main layers:

- Group of frequency-planes or group of frames layer,

- Frequency-plane layer,

It can be seen that the total amount of overhead information is quite insignificant. As for example, with the number of video frames in a GOF $L = 8$ and CIF (352×288) frame resolution, the overhead information consumes only around 2% of the total bit budget at coding rate of 0.01 bits per pixel.

"Bit allocation" is the most important issue that plays a crucial role especially at very low bit-rate coding scenario. Because, any unwise bit distribution may result in severe degradation of image/video quality. To solve such a critical problem, we have followed a two-step-approach. With the total amount of bit budget to encode a GOF in our hand, we first distribute the amount among different frequency planes. Next, we go for allocating the bit-budget for a particular frequency plane among different subbands.

For sake of discussion, let us assume that the total available bit budget to encode a GOF is $\tilde{T}$. We, first of all, subtract the amount of bits required by the overhead information (please refer to Figure 2.15 for details of the associated overhead information) and obtain the amount actually available for encoding purpose. Let us say, this amount is T. Now, if we assume that $x_1, x_2, \ldots x_L$ are L (L is the number of video-frames in a GOF, also the number of frequency planes after applying DCT) random variables such that the coefficient values of the DC plane are the sample values of the first random variable x_1, coefficient values of the first AC plane are the sample values of the second random variable x_2 and so on and also if the variances of these random variables (in other words of the frequency planes) be σ_1^2 , σ_2^2 , ... σ_L^2, then using the method described in [17], we can calculate the average number of bits per coefficient in i^{th} frequency plane as

$$R_i = R + \frac{1}{2}\log_2 \frac{\sigma_i^2}{\prod_{k=1}^{L} (\sigma_k^2)^{\frac{1}{L}}} \; ; \quad i = 1 \cdots L \tag{2.5}$$

where R is the average number of bits per pixel and is obtained as

$$R = \frac{T}{XYL} \tag{2.6}$$

Here, X is the number of lines in each frequency plane, Y is the number of DCT coefficients in every line. Hence, the total number of bits to be used for i^{th} frequency plane becomes

$$T_i = R_i XY \tag{2.7}$$

Noteworthy, that the constraint

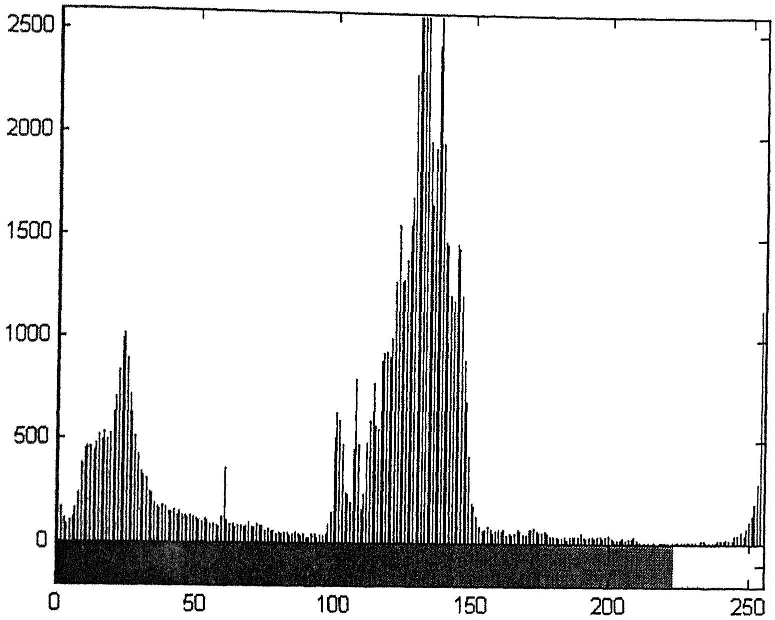
$$\sum_{i=1}^{L} T_i = T \tag{2.8}$$

also gets satisfied by Equations (2.5), (2.6) and (2.7).

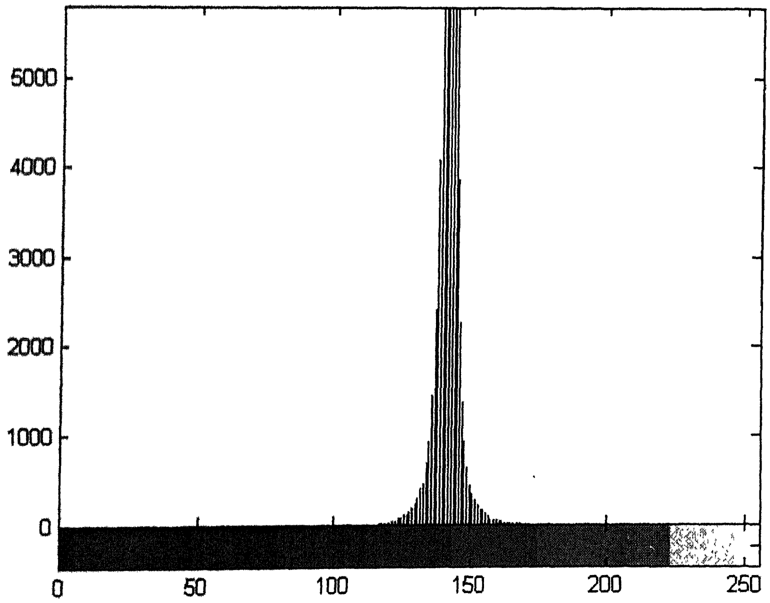
Unfortunately, we can not use Equation (2.5) to solve the bit allocation problem for any GOF. To be precise, the underlying assumption in the formulation of Equation (2.5) is that the rate-distortion function should be same for all the random variables which in turn is satisfied when probability density function (pdf) of all the random variables be similar, and the same quantizer model is used for all. So, before applying Equation (2.5), we must ensure that these conditions are fulfilled.

While the quantizer model may be assumed to be same for all, not all the random variables are guaranteed to have similar pdf. In fact, the pdf of x_1 is significantly different from those of the remaining random variables x_2, ... x_L (refer to Figure 2.16), particularly when DC frequency plane itself is used for subsequent spatial decorrelation. Therefore, for the GOFs, in which DC frequency plane is chosen by the decision maker, a different strategy needs to be used to allocate the total bit budget. In our present work, we have followed some heuristic approach. We simply perform distribution of bits among different frequency planes in proportion to their standard deviations such that sum of the amounts for individual frequency planes equals T.

But, in the cases when the decision maker selects the residue for further processing, the pdf constraint can also be satisfied. This is so because, just like the higher frequency planes, the residue contains huge number of zero/low amplitude pixels and the histogram plots of all of them look alike (similar to Figure 2.16(b)). Still, the solution can not be perfectly achieved. Because, at very low bit rate (<1bpp) the rate-distortion model that is used to formulate Equation (2.5) does not fit well and



(a) Histogram of the DC plane that is obtained from the 1^{st} GOF of 'Claire' sequence



(b) Histogram of one of the AC planes that are obtained from the 1^{st} GOF of 'Claire' sequence

Figure 2.16: Example Histograms for Different Frequency Planes

values of R_i's often become negative. Many approaches have been used in the past to avoid this awkward situation [17]. One of them is to set the negative R_is to zero. But, this increases the average bit rate above the constraint. Therefore, next the non-zero R_is are uniformly reduced until the average bits per pixel becomes equal to R.

Here, we suggest an alternative approach. We make a small modification in Equation (2.5) so as to ensure that R_is take positive values.

Suppose, the average bits per pixel value required is R and the Equation (2.5) results in some negative R_is. We go on increasing the average bits per pixel value as $\tilde{R} = rR$ where $r = 2, 3, \ldots$ and at each step i.e. for each value of r we compute $\tilde{R}_i$s using

$$\tilde{R}_i = \tilde{R} + \frac{1}{2}\log_2 \frac{\sigma_i^2}{\prod_{k=1}^{L} (\sigma_k^2)^{\frac{1}{L}}} \; ; \quad i = 1 \cdots L \tag{2.9}$$

We stop at that point when the values of $\tilde{R}_i$s all become positive. Say, the corresponding value of r be r'. Finally, to obtain the values of R_is we divide $\tilde{R}_i$s by r' i.e.

$$R_i = \frac{1}{r'}\tilde{R}_i = \frac{1}{r'}\tilde{R} + \frac{1}{2r'}\log_2 \frac{\sigma_i^2}{\prod_{k=1}^{L} (\sigma_k^2)^{\frac{1}{L}}}; \quad i = 1 \cdots L \tag{2.10}$$

Note that, the constraint in Equation (2.8) is also satisfied using the above method. We have tried with all the available methods (including ours). Ultimately, we have adopted our own approach as it showed improved performance over others.

Equations (2.5), (2.9) and (2.10) can also be modified to include the psycho-visual importance of different frequency planes. In order to do that we require to use the weighted variances of the frequency planes such that Equations (2.5), (2.9) and (2.10) become modified as

$$R_i = R + \frac{1}{2}\log_2 \frac{w_i \, \sigma_i^2}{\prod_{k=1}^{L} (w_i \, \sigma_k^2)^{\frac{1}{L}}} \; ; \quad i = 1 \cdots L \tag{2.11}$$

$$\tilde{R}_i = \tilde{R} + \frac{1}{2}\log_2 \frac{w_i \, \sigma_i^2}{\prod_{k=1}^{L} (w_i \, \sigma_k^2)^{\frac{1}{L}}} \; ; \quad i = 1 \cdots L \tag{2.12}$$

Weights	Values
w_1	0.40
w_2	0.20
w_3	0.09
w_4	0.08
w_5	0.07
w_6	0.06
w_7	0.05
w_8	0.05

Table 2.1: Psycho-Visual Weights For Different Frequency Planes

$$R_i = \frac{1}{r'}\tilde{R}_i = \frac{1}{r'}\tilde{R} + \frac{1}{2r'}\log_2 \frac{w_i\,\sigma_i^2}{\prod\limits_{k=1}^{L}(w_i\,\sigma_k^2)^{\frac{1}{L}}}; \quad i = 1 \cdots L \tag{2.13}$$

Here $\{w_i\}$ is the set of psycho-visual weights that satisfies the following constraint.

$$\sum_{i=1}^{L} w_i = 1 \tag{2.14}$$

The values of w_is used in our experiment are listed in Table 2.1.

Once, we know the total amount of bits to be used to code a particular frequency plane, the remaining part of the bit allocation problem (i.e. bit allocation among different spatial subband coefficients) can be efficiently handled with the help of SPIHT algorithm (discussed in section 2.4.3).

2.6 Simulation result

In this section, we show the performance of our codec. Experiments are conducted on the standard videos like "Akiyo", "Claire", "Miss America", "Salesman", "Hall Monitor". The first three "head & shoulder" type videos are characterized by small object motion and stationary background. The fourth video is also of similar kind with multiple motions. "Hall Monitor" is particularly suitable for monitoring applications, where the background is always fixed and sometimes some persons/objects appear and disappear. Each of these videos have CIF (352×288) frame resolution with frame rate of 30 frames/sec.

Peak signal-to-noise ratio (PSNR) calculated as

$$PSNR_{dB} = 10\log_{10}\left(\frac{255^2}{MSE}\right) \tag{2.15}$$

where

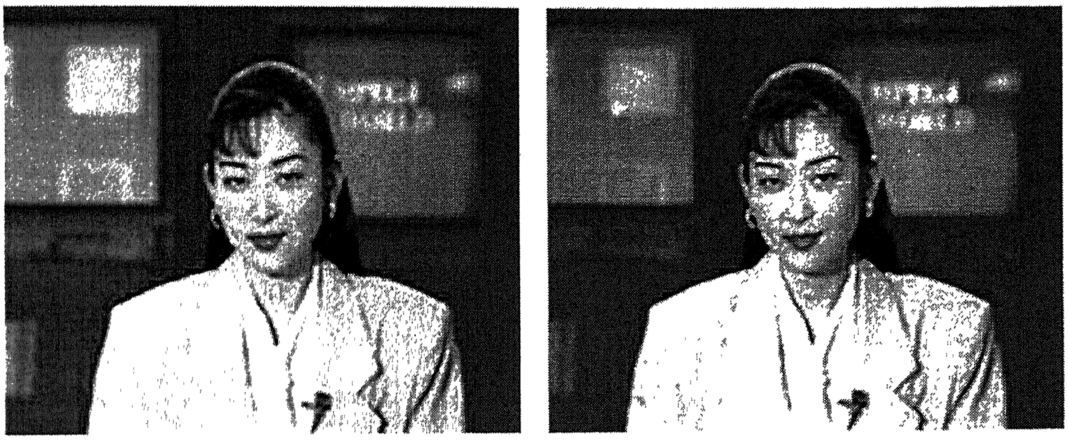
$$MSE = \frac{1}{NM}\sum_{n=1}^{N}\sum_{m=1}^{M}\left(Frame_o\left[n,m\right] - Frame_r\left[n,m\right]\right)^2 \tag{2.16}$$

has been used as an objective measure of the frame quality. In Equation (2.16), $Frame_o$ stands for an original frame and $Frame_r$ denotes corresponding reconstructed frame. Although we emphasize that the true subjective quality of a reconstructed video-frame does not have any rigid connection with the frame-PSNR. Also, instead of inspecting the quality frame-by-frame, we should judge the video-quality as a whole, treating video as a three-dimensional entity. Over and above, the actual evaluation of the reconstructed video quality ought to be made by playing back the video with associated sound!

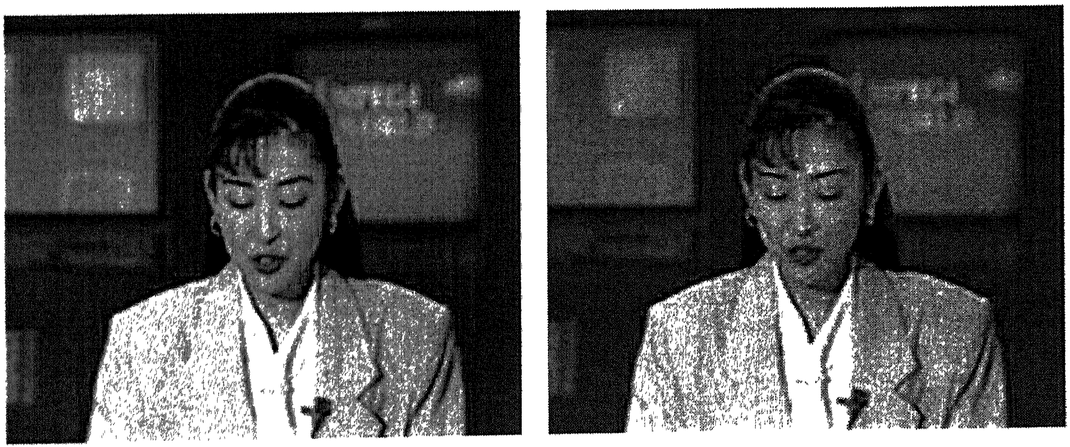
Figures 2.17, 2.18, 2.19, 2.20, 2.21, 2.22, 2.23, 2.24 display few of the results obtained using the proposed codec at bit-rate $121.65Kbps$ (compression ratio: 200:1). Figure 2.25 shows the result obtained for "Salesman" sequence at a bit-rate $152Kbps$ (compression ratio: 160:1). Respective variations in frame-PSNR are shown in Figures 2.26, 2.27, 2.28, 2.29, and 2.30. It can be observed that the PSNR deteriorates rapidly at the portions where the motion activity soars high. One reason behind this is, when motion activity increases, the amount of information to be conveyed to the decoder for successful decoding also increases. This in turn puts more demand on the bit-budget, which we have not provided. Rather, throughout the entire sequence, we have kept the total bit budget per GOF constant. Another reason may be that in our actual simulation, we have not really implemented the "Decision Maker" part of the encoder. We have always chosen the residue for further processing. Thus, when the motion activity in the video increases, the effect of insufficient residue coding keeps on multiplying. However, during the low-motion-period, the quality can again be recovered. The point that is worth mentioning is that given the total bit-budget per GOF as T, we have used double of this, i.e. $2T$ to code the first GOF. This is quite reasonable. Because, for the first GOF, we have to code the absolute information. Whereas for other GOFs we code the relative information. Hence, the first GOF requires greater bit-budget.

Next, we have compared some of our results with those obtained using 3D-SPIHT [9] proposed by Kim and Pearlman. The proposed codec has been noticed to yield better/comparable subjective quality at different low bit rates. Figures 2.31, 2.32 demonstrate the comparison results. One of the most objectionable drawbacks of 3D-SPIHT reported is that the decoded video when played back appears irritating due to the presence of temporal jerk. However, the codec developed by us is able to retain the temporal smoothness in the reconstructed video. The DPCM-like coding algorithm for encoding the DC planes may be the reason for the absence of jerk.

Finally, we emphasize that no post-processing (filtering) has been carried out to incorporate visually pleasant soothing effect. In fact, any unwise filtering can eat away important details, thereby destroying useful informations.

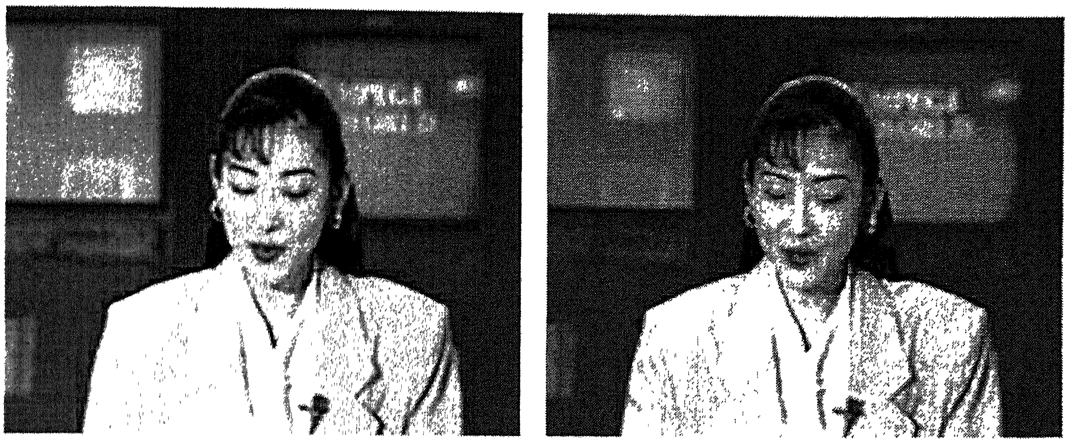


(a) Original 9th frame (b) Reconstructed 9th frame : bpp 0.04

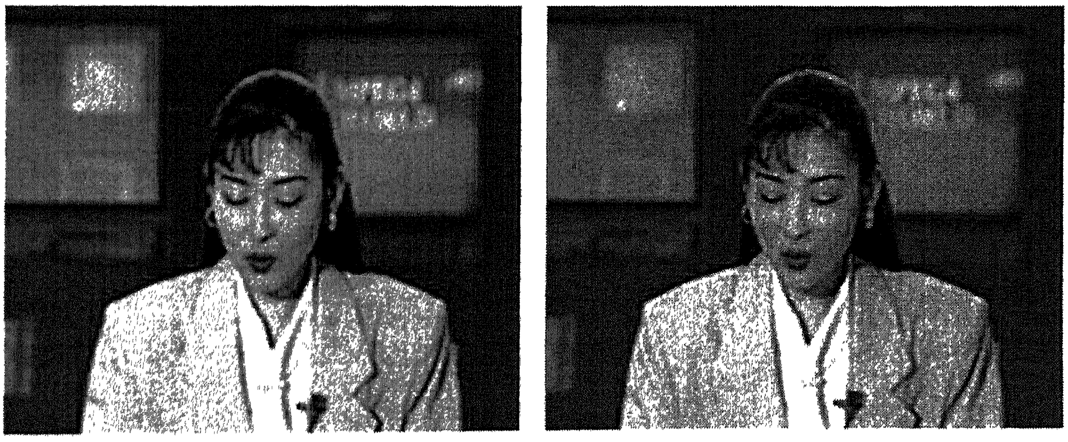


(a) Original 40th frame (b) Reconstructed 40th frame : bpp 0.04

Figure 2.17: Akiyo : Coded at 122 Kbps (Compression Ratio 200:1)

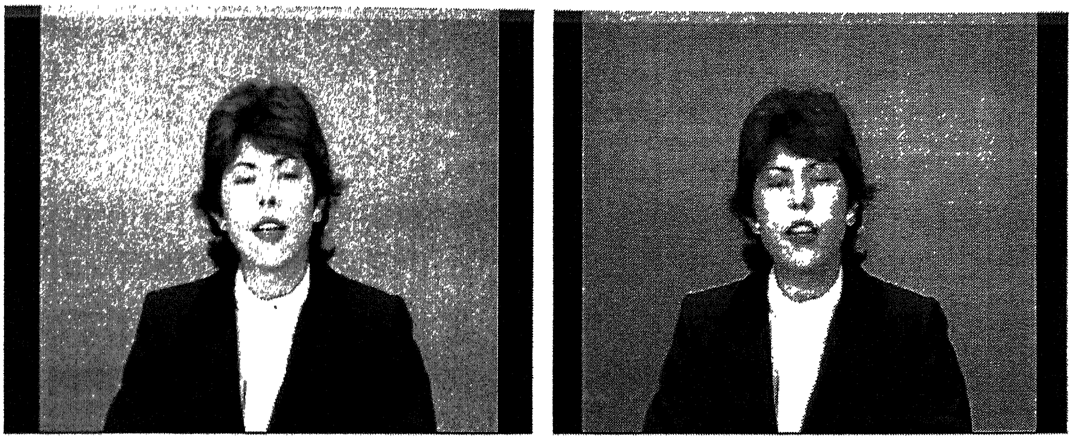


(a) Original 80th frame (b) Reconstructed 80 th frame : bpp 0.04



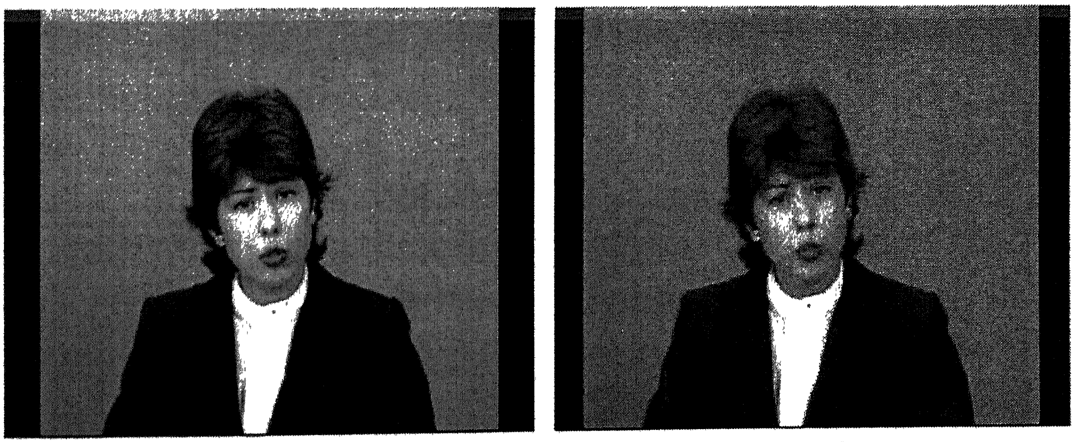
(a) Original 121st frame (b) Reconstructed 121st frame : bpp 0.04

Figure 2.18: Akiyo : Coded at 122 Kbps (Compression Ratio 200:1)



(a) Original 11th frame

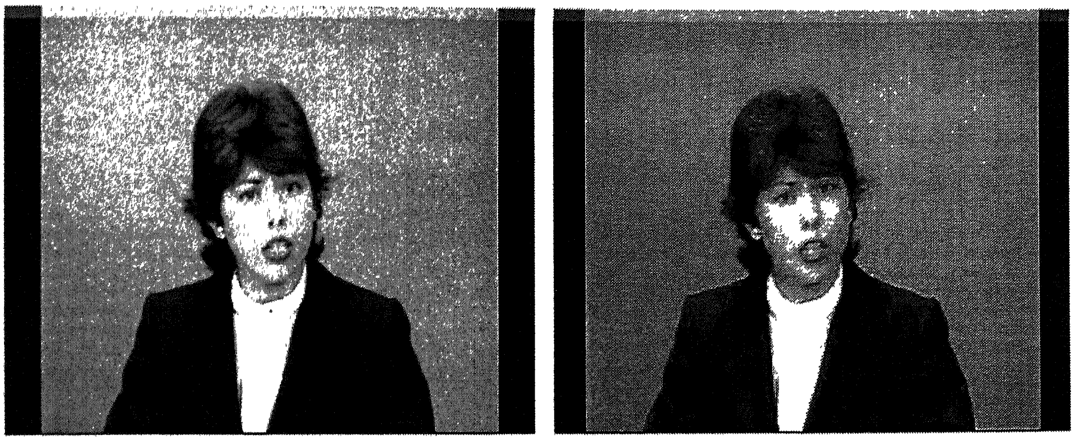
(b) Reconstructed 11th frame : bpp 0.04



(a) Original 62nd frame

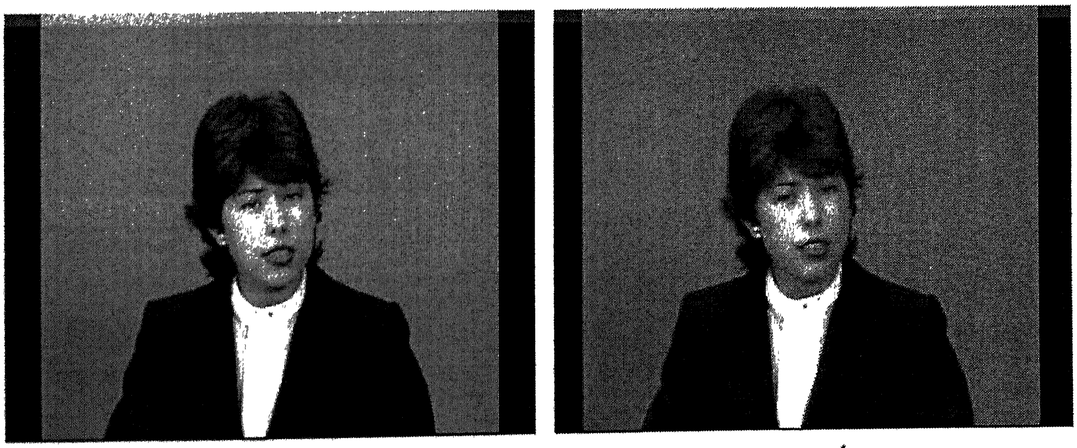
(b) Reconstructed 62nd frame : bpp 0.04

Figure 2.19: Claire : Coded at 122 Kbps (Compression Ratio 200:1)



(a) Original 113th frame

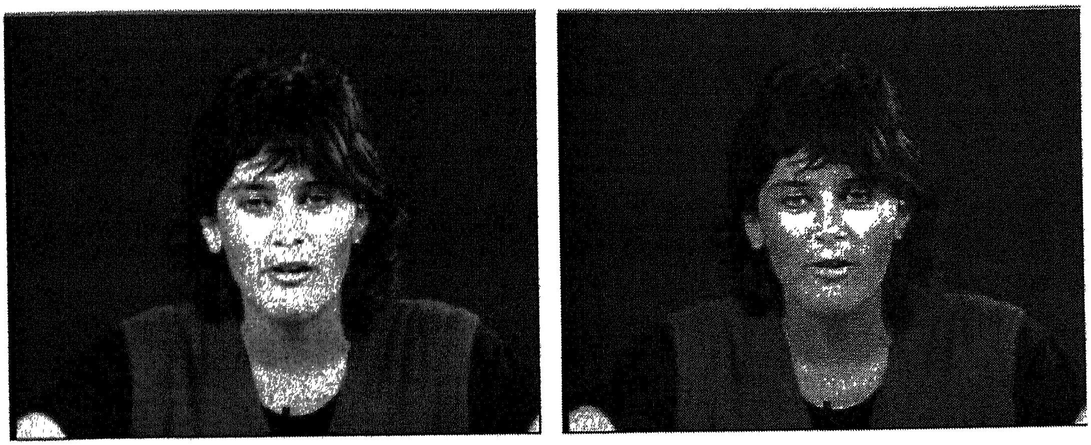
(b) Reconstructed 113th frame : bpp 0.04



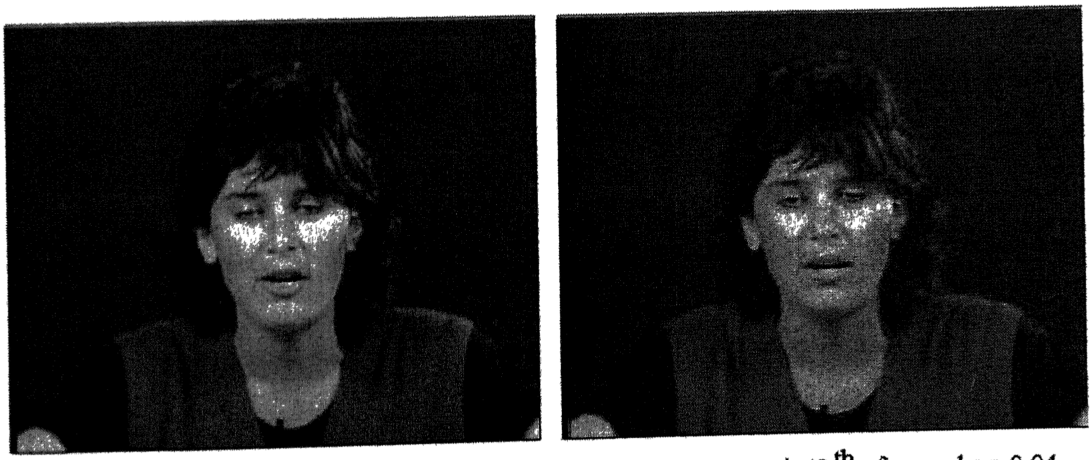
(a) Original 151st frame

(b) Reconstructed 151st frame : bpp 0.04

Figure 2.20: Claire : Coded at 122 Kbps (Compression Ratio 200:1)

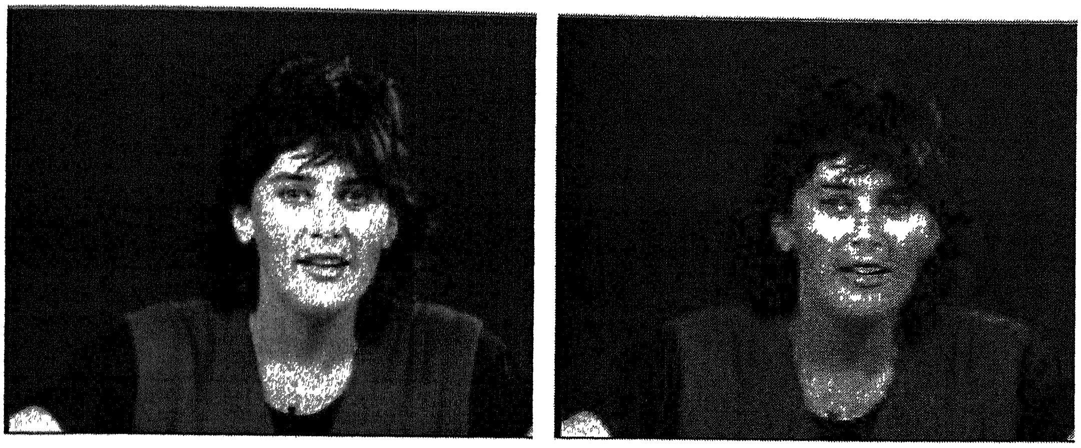


(a) Original 12th frame (b) Reconstructed 12th frame : bpp 0.04

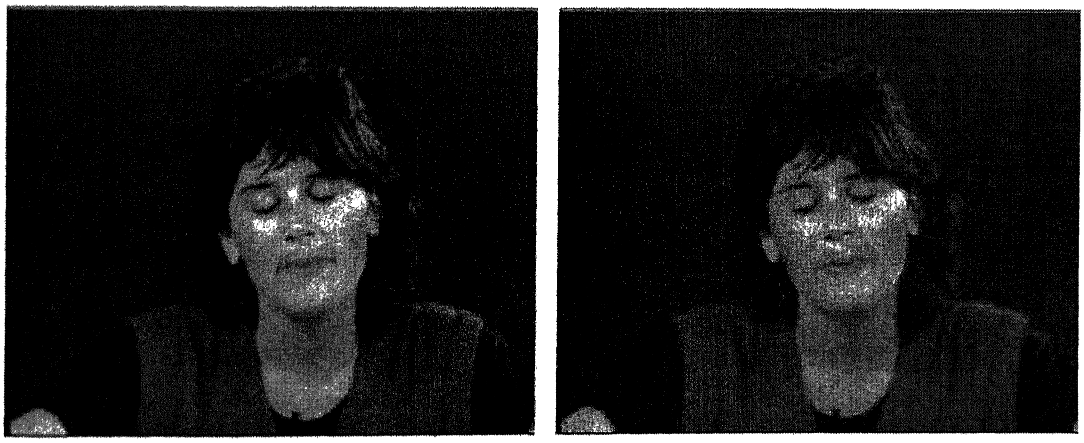


(a) Original 49th frame (b) Reconstructed 49th frame : bpp 0.04

Figure 2.21: Miss America : Coded at 122 Kbps (Compression Ratio 200:1)

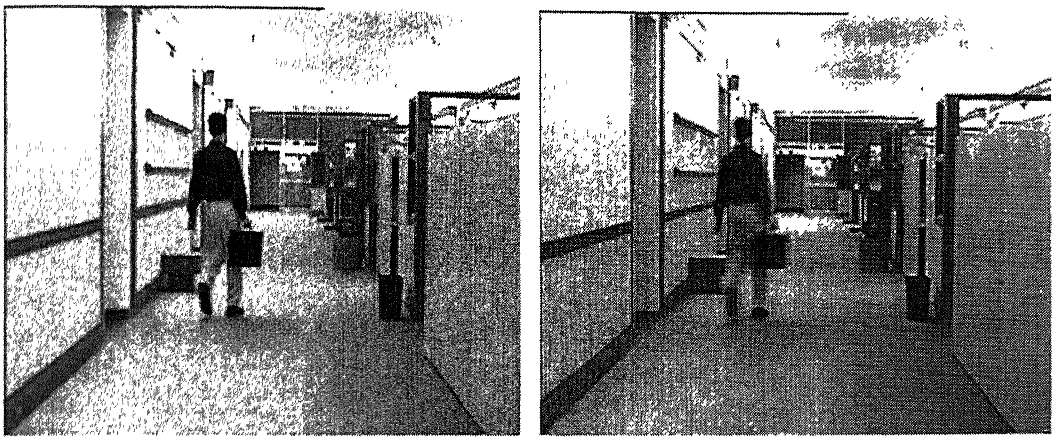


(a) Original 81st frame (b) Reconstructed 81st frame : bpp 0.04

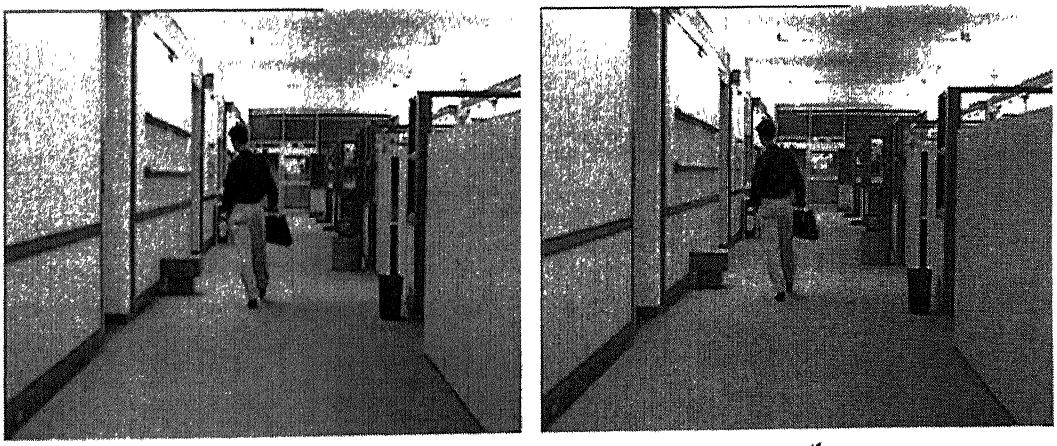


(a) Original 121st frame (b) Reconstructed 121st frame : bpp 0.04

Figure 2.22: Miss America : Coded at 122 Kbps (Compression Ratio 200:1)

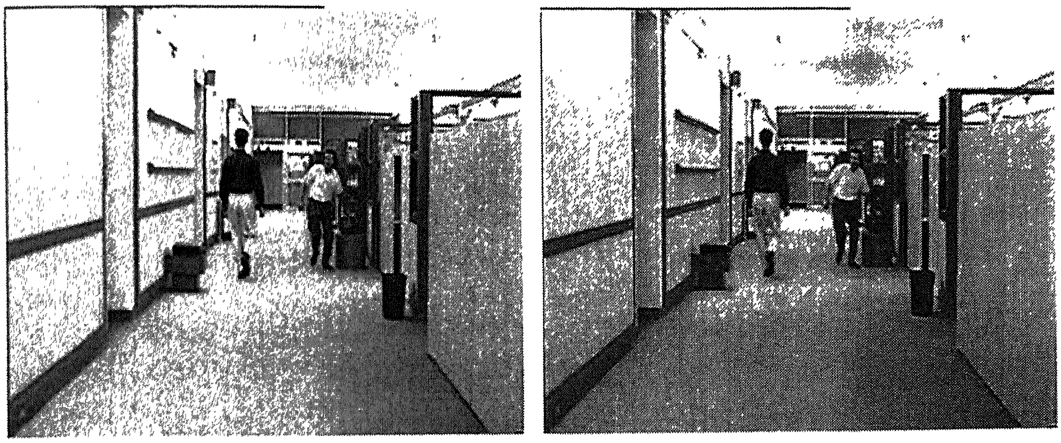


(a) Original 50th frame (b) Reconstructed 50th frame : bpp 0.04

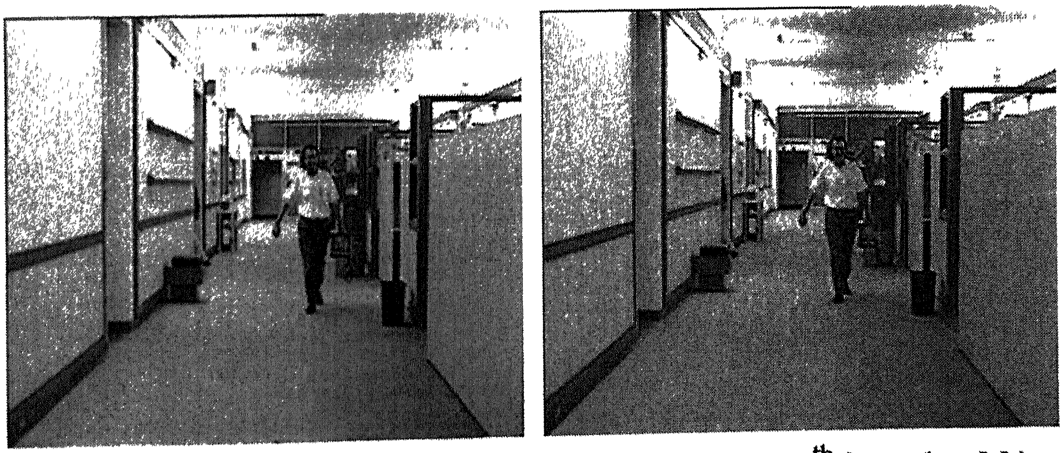


(a) Original 74th frame (b) Reconstructed 74th frame : bpp 0.04

Figure 2.23: Hall Monitor : Coded at 122 Kbps (Compression Ratio 200:1)

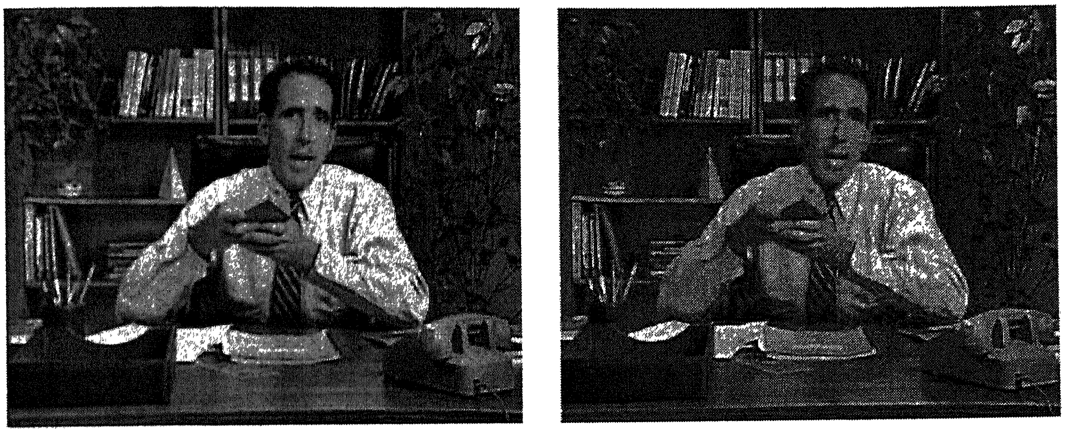


(a) Original 170th frame (b) Reconstructed 170th frame : bpp 0.04



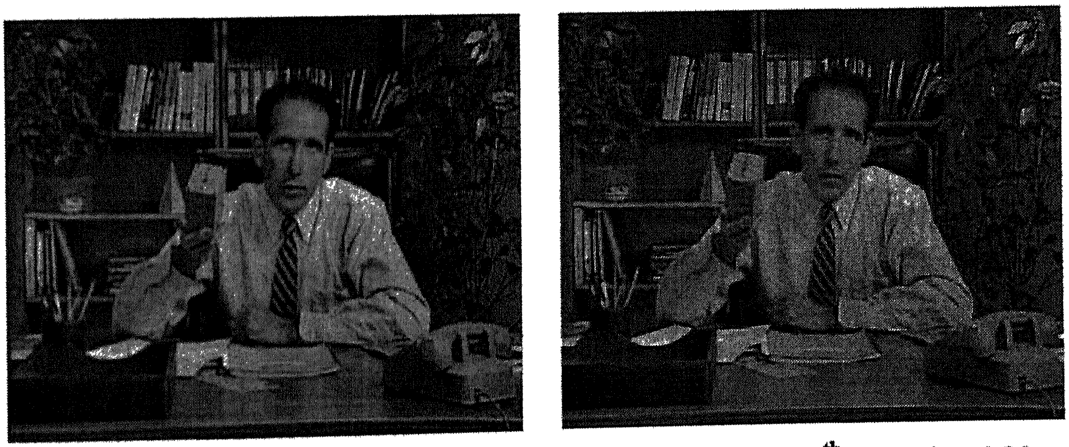
(a) Original 248th frame (b) Reconstructed 248th frame : bpp 0.04

Figure 2.24: Hall Monitor : Coded at 122 Kbps (Compression Ratio 200:1)



(a) Original 19th frame

(b) Reconstructed 19th frame : bpp 0.05



(a) Original 76th frame

(b) Reconstructed 76th frame : bpp 0.05

Figure 2.25: Salesman : Coded at 152 Kbps (Compression Ratio 160:1)

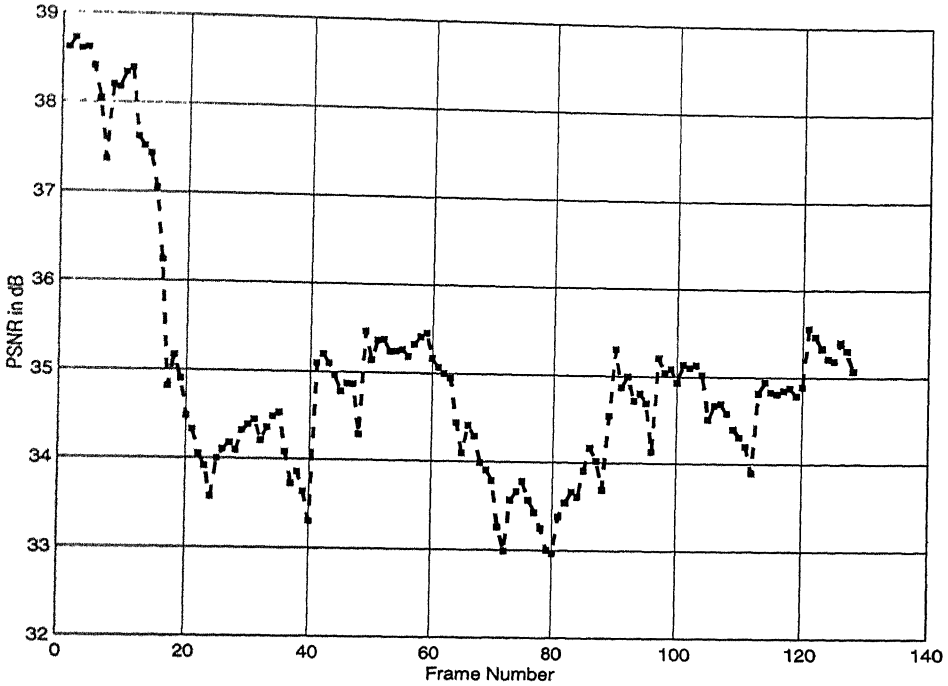


Figure 2.26: Variations in Frame-PSNR for 'Akiyo' Sequence at 122Kbps

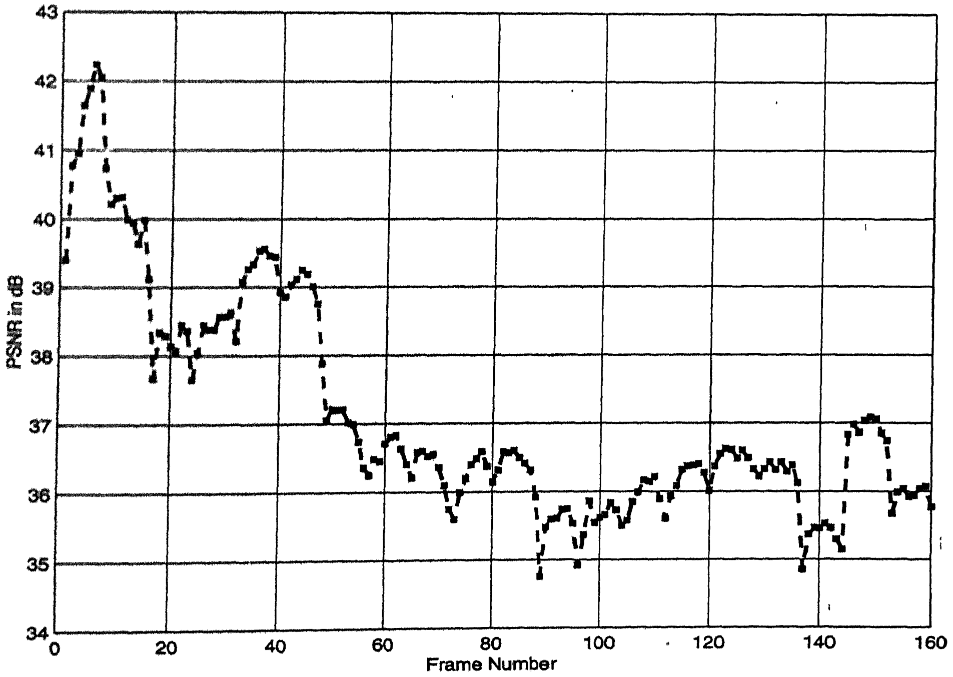


Figure 2.27: Variations in Frame-PSNR for 'Claire' Sequence at 122Kbps

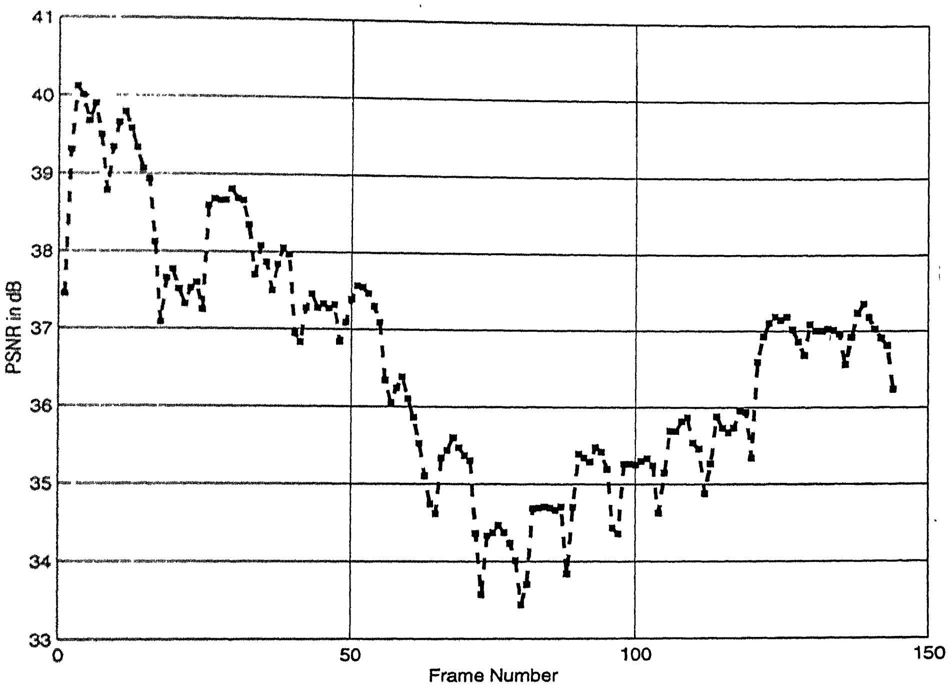


Figure 2.28: Variations in Frame-PSNR for 'Miss America' Sequence at 122Kbps

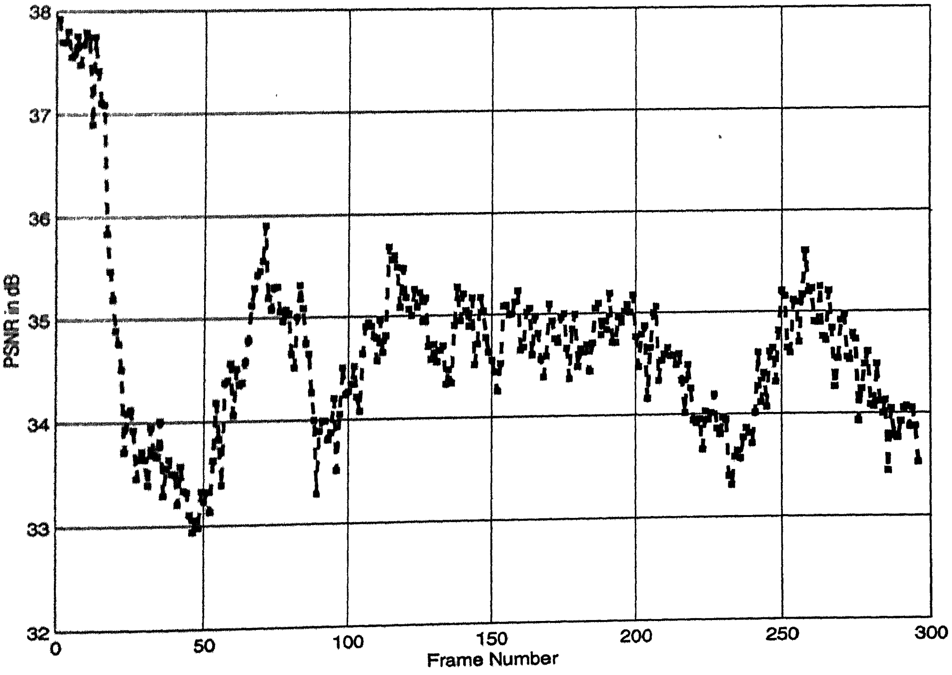


Figure 2.29: Variations in Frame-PSNR for 'Hall Monitor' Sequence at 122Kbps

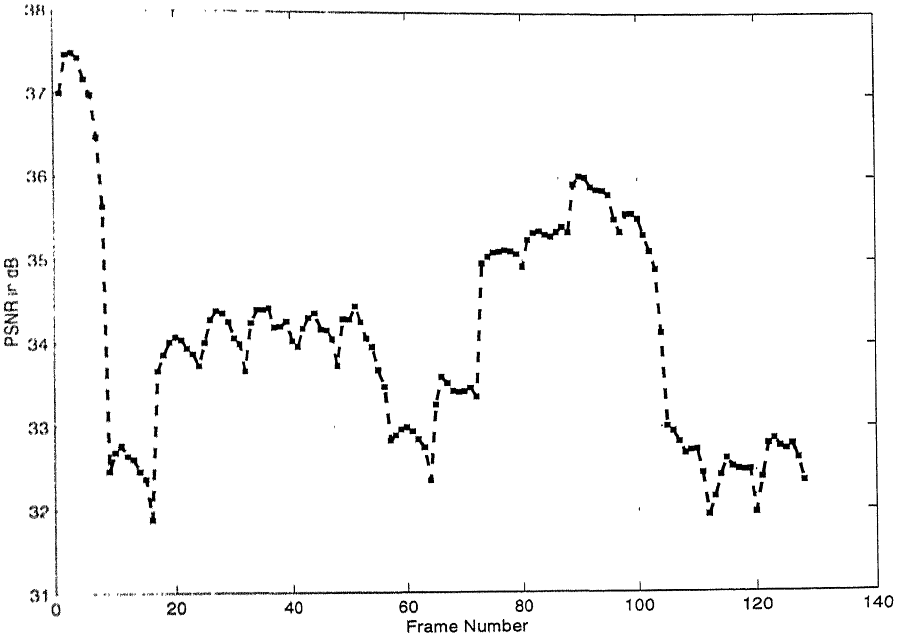


Figure 2.30: Variation in Frame-PSNR for "Salesman" Video Sequence at 152Kbps

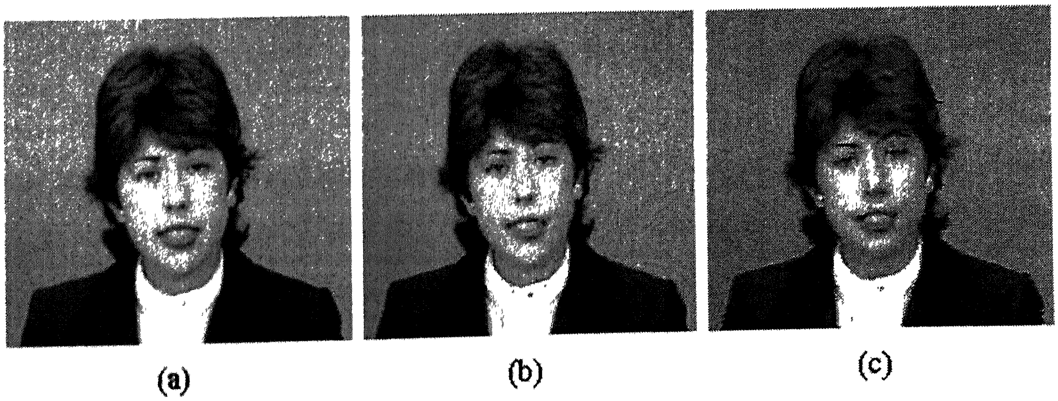

(a) (b) (c)

Figure 2.31: Performance Comparison at 0.04bpp : (a) Original 32^{rd} Frame of 'Claire', (b) Reconstructed Frame Using the Proposed Codec, (c) Reconstructed Frame Using 3DSPIHT

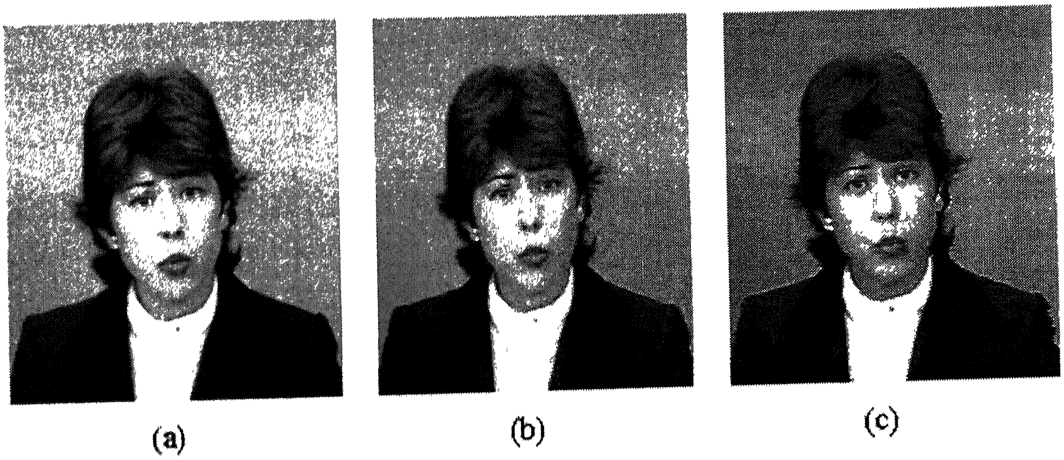


Figure 2.32: Performance Comparison at 0.04bpp : (a) Original 46th Frame of 'Claire', (b) Reconstructed Frame Using the Proposed Codec, (c) Reconstructed Frame Using 3DSPIHT

Chapter 3

VLBR Video Coding: An Improved Codec

The codec proposed in the previous chapter succeeds to retain overall good visual quality at bit-rate around 0.04 bpp. However, the reconstructed frame quality has been noticed to deteriorate when the motion activity as a whole increases. Also around 0.02 bpp, significant distortion has been observed at the regions having larger motion. In this chapter, we suggest one simple remedy to overcome this problem. Experimental results have been shown to demonstrate the improvement achieved.

3.1 Problem identification

The main reason behind the frame quality degradation during greater motion activity in a video sequence is the inadequate amount of bit budget. It precludes the codec from coding the temporal details to sufficient extent. Therefore, we need to create some framework that will help the codec to spend more bits for coding high-motion-parts and fewer bits in coding static background or objects. Before directly getting into the solution, let us see, how the bits are wasted in the process of encoding. Figure 3.1 shows the set of first four frequency planes that are obtained after applying DCT along the time dimension of the first GOF of video "Claire". Consider the third

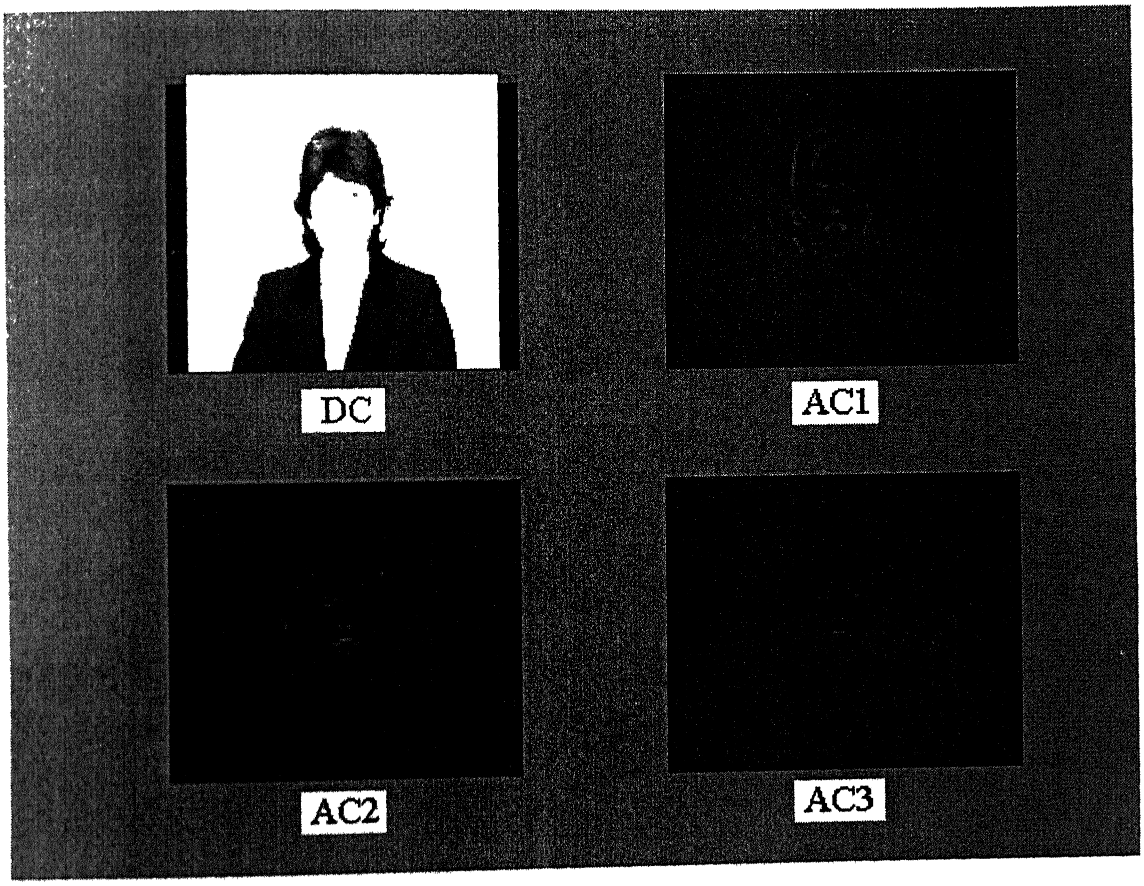


Figure 3.1: First Four Frequency Planes Obtained After Temporal Decorrelation of the First GOF of 'Claire'

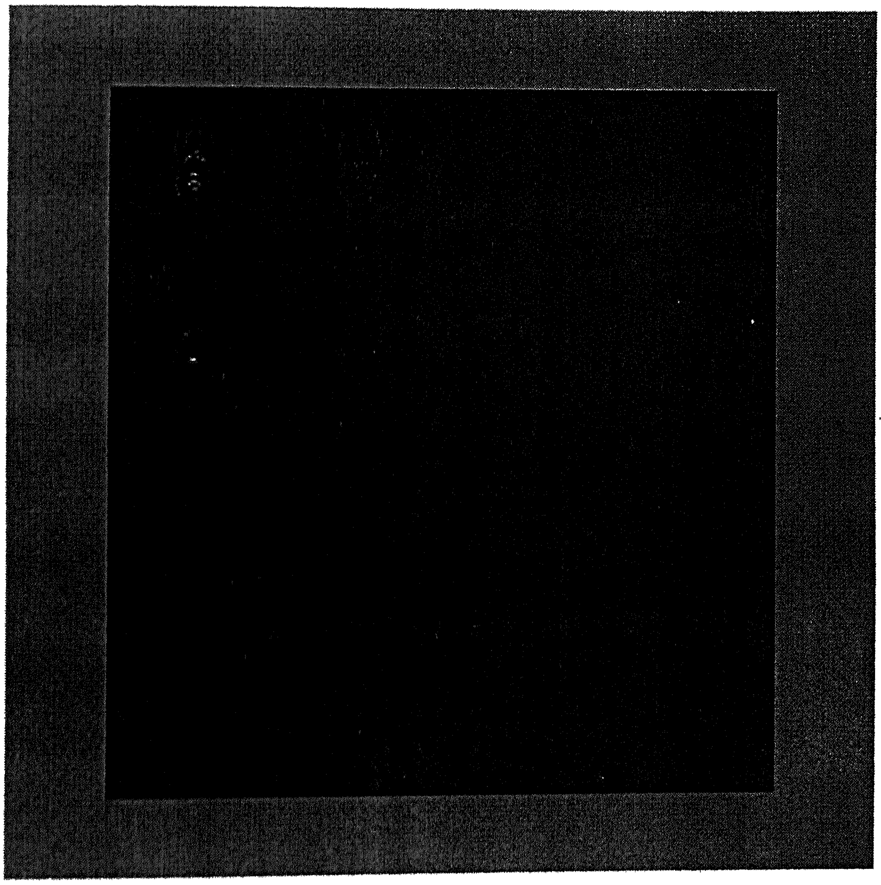


Figure 3.2: Two Levels of Wavelet Decomposition of the Third Frequency Plane

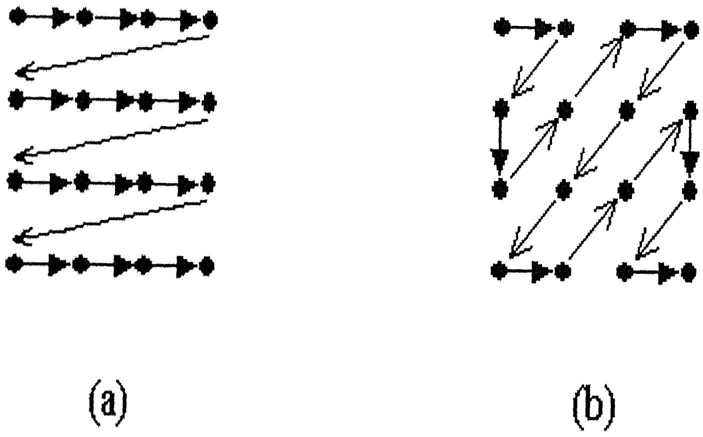


Figure 3.3: Frequently Used Scanning Patterns

frequency plane, i.e. AC2. Its wavelet transform (with 2 levels of decomposition) has been shown in Figure 3.2. SPIHT algorithm is applied to quantize and code this transformed frequency plane. In actual encoding process, which indeed is accomplished using multiple scans through the wavelet coefficients, a threshold level (at each pass, this value is reduced to its half) is set depending on the maximum magnitude of the wavelet coefficients. Three lists are maintained meanwhile : LIP, LIS & LSP. At each pass, the elements in LIP, LIS are tested against the current threshold for significance. If any of them is found to be not significant, a "0" bit is sent to indicate the event "insignificant", to the decoder. Now, in the scanning process, if all the insignificant coefficients are encountered first, many bits will be consumed only to inform the decoder that the coefficients are not significant! And few bits will remain to code the values that are of higher magnitudes and hence likely to have greater influence on the decoded quality! As in Figure 3.2, following either of the scanning patterns shown in Figure 3.3 the insignificant coefficients occurring at the topmost left corner can not be avoided and

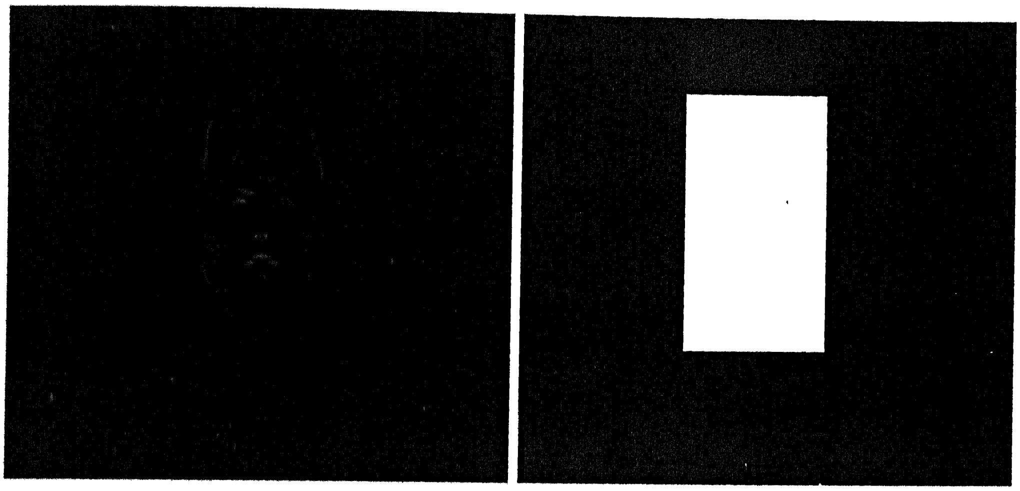

(a) (b)

Figure 3.4: (a) The Second AC Plane, (b) The Mask

therefore considerable amount of bits have to be used for conveying their insignificance to the decoder.

3.2 A possible solution

In order to avoid the problem discussed in section 3.1, we reduce the area over which wavelet decomposition and subsequent SPIHT algorithm are to be applied. For example, we choose only the central part of the third frequency plane (as indicated by the mask shown in Figure 3.4) for purpose of wavelet coding. Note that, this central part contains most of the significant coefficients. As the motion increases, temporal frequency also rises. And as the temporal frequency increases, the magnitude of the coefficients in the higher frequency planes also increases. Thus, by selecting only the significant portion of any AC plane for further processing, we can indirectly utilize more number of bits from the available bit-budget for coding the regions containing larger movements.

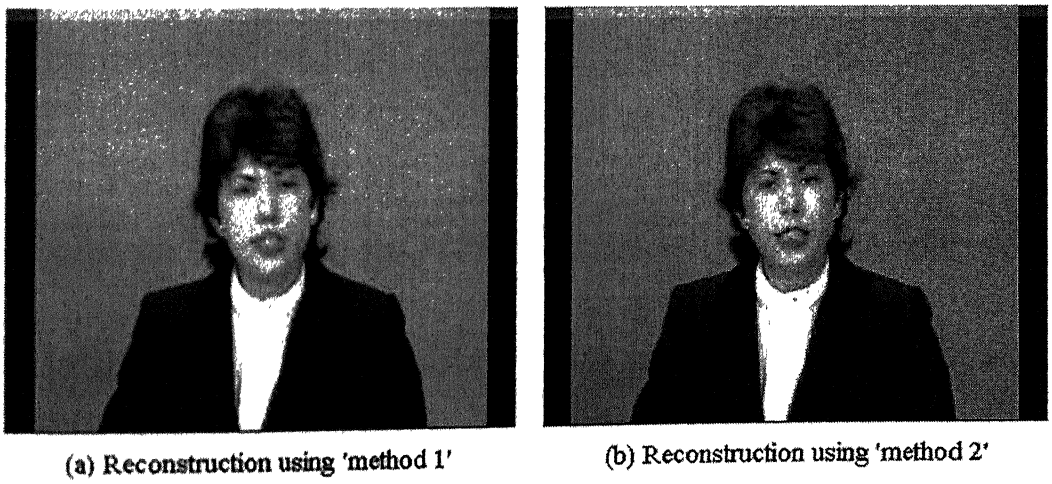
पुरुषोत्तम काशीनाथ केलकर पुस्तकालय
भारतीय प्रौद्योगिकी संस्थान कानपुर
अवाप्ति क्र० A 144427

3.3 Implementation Issue

Now, in choosing the area containing the significant wavelet coefficient values, we need to generate a mask for each of the frequency planes. For that, we select thresholds that will be different for different frequency planes. (For example, in our experiment, we have set a threshold value that is equal to 10% of the peak). We compare the wavelet coefficients of a particular frequency plane with the corresponding threshold and produce (say) "1" in case the coefficient value is greater than or equal to the threshold. Otherwise we produce (say) "0". This comparison will eventually give rise to a binary image containing "1"s and "0"s, which helps us to generate the mask. The binary image will actually contain a large number of small, unconnected spurious regions/points (containing "1") in addition to the region of our interest. Those unwanted regions are simply eliminated. (In our experiment, any regions whose boundary length is less than 50 pixels have been straightway neglected). Finally, we enclose the remaining "1"s with a smallest enclosing rectangle and thus mark the portion that can be coded in later phase. In this way, we create the masks for each individual frequency planes. It is worth noting that, to assure proper decoding at the other end, we need to transmit the coordinates of one of the four corners of the enclosing rectangle and the lengths of its two perpendicular arms. Assuming that each of these additional information requires one byte, we have to spend additional 32 bits. However, this amount is presumably much less than that required to convey the "insignificant" events (as described earlier) to the decoder end. Still, if we insist not to waste bits in this manner, we can also use a common mask that encloses the significant regions of all the frequency planes and send the additional side information only once. In our experiment, we have followed the second approach. We have used a common mask and also the lengths of the arms of the rectangle had been chosen out of the set {64, 96, 128, 160, 192, 256}. The reason behind this is to allow more levels of wavelet decomposition in subsequent stage, which in turn, helps in achieving higher compression.

3.4 Simulation result

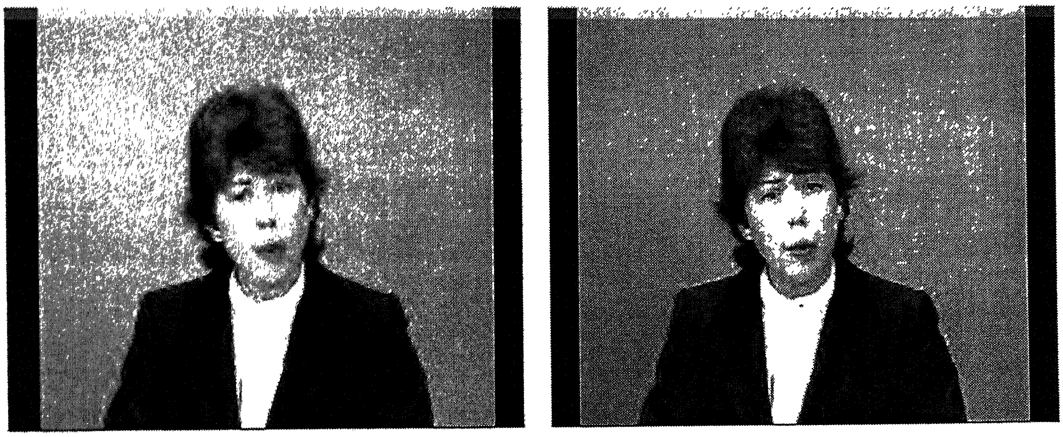
In this section we show the improvement achieved in the performance. Frames on the left are obtained using the coding scheme described in the previous chapter (let us call it 'method 1') and those in the right are obtained applying the suggested modification (let us call it 'method 2').



(a) Reconstruction using 'method 1'

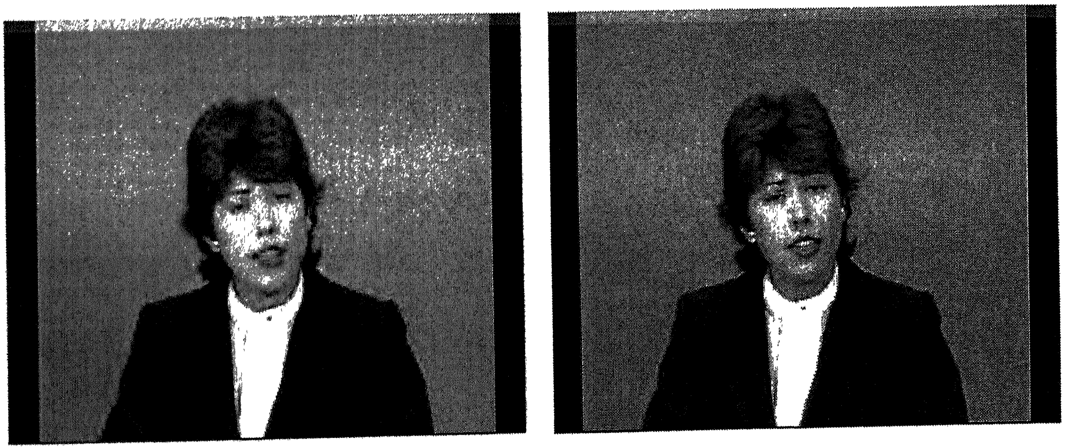
(b) Reconstruction using 'method 2'

Figure 3.5: 32^{nd} Frame of 'Claire' : Coded at 60.8Kbps (Compression Ratio 400:1)



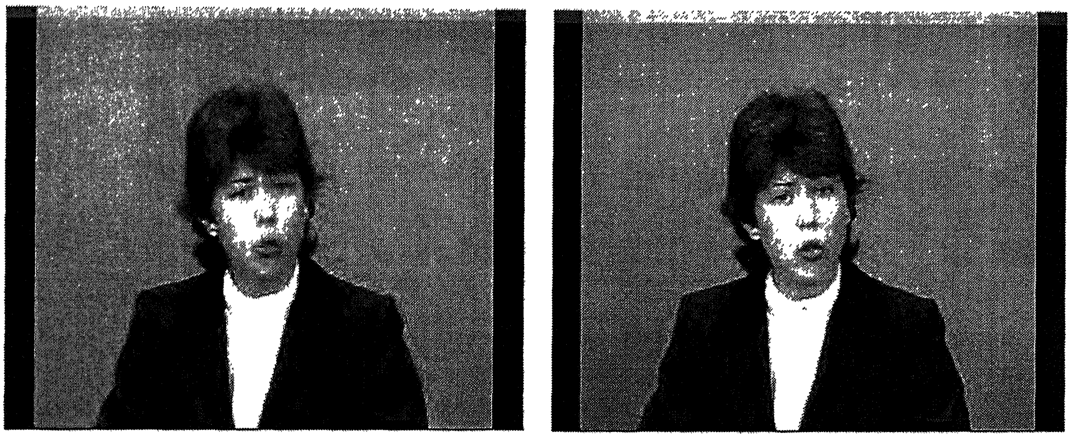
(a) Reconstruction using 'method 1' (b) Reconstruction using 'method 2'

Figure 3.6: 70th Frame of 'Claire' : Coded at 60.8Kbps (Compression Ratio 400:1)



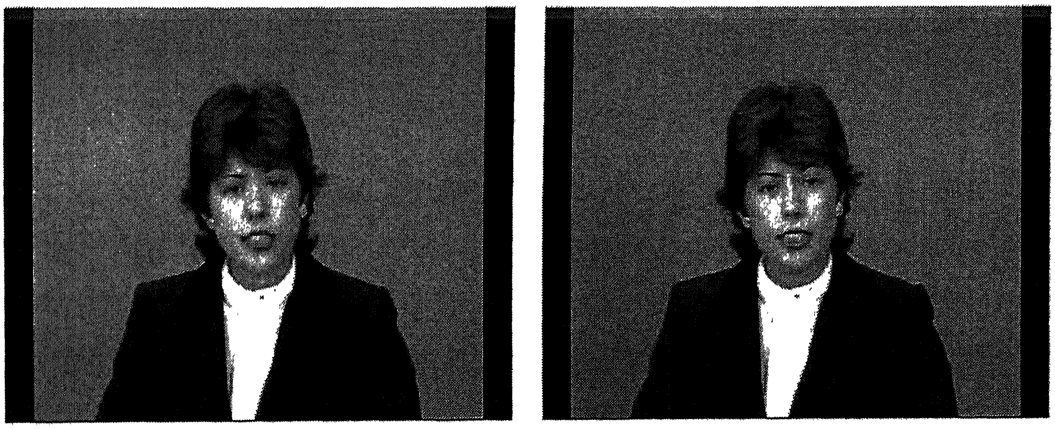
(a) Reconstruction using 'method 1' (b) Reconstruction using 'method 2'

Figure 3.7: 88th Frame of 'Claire' : Coded at 60.8Kbps (Compression Ratio 400:1)



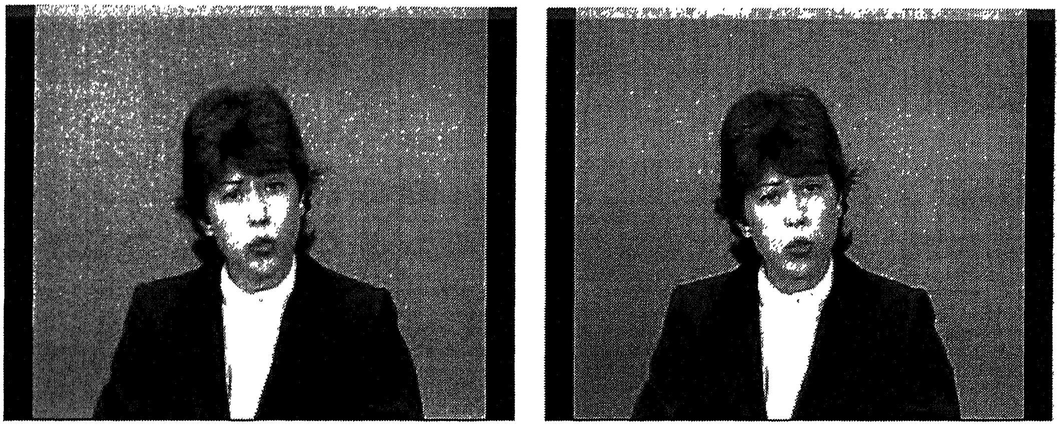
(a) Reconstruction using 'method 1' (b) Reconstruction using 'method 2'

Figure 3.8: 124^{th} Frame of 'Claire' : Coded at 60.8Kbps (Compression Ratio 400:1)



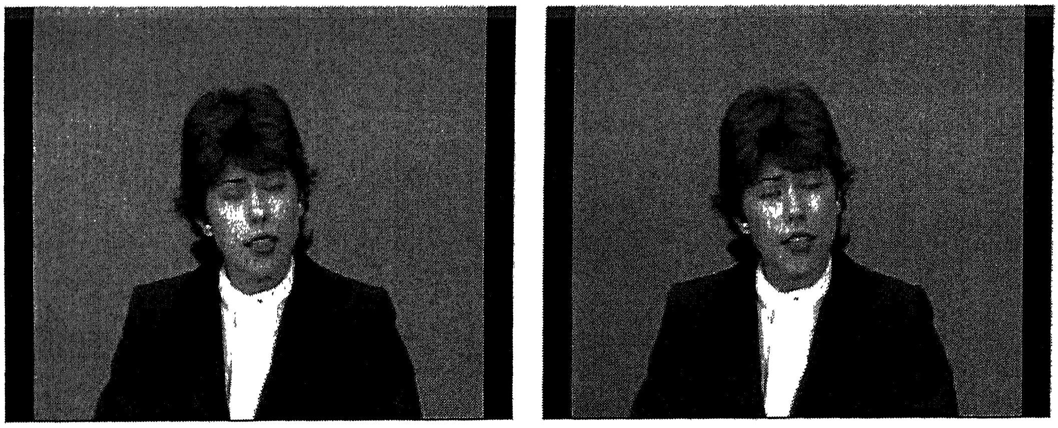
(a) Reconstruction using 'method 1' (b) Reconstruction using 'method 2'

Figure 3.9: 31^{st} Frame of 'Claire' : Coded at 122Kbps (Compression Ratio 200:1)



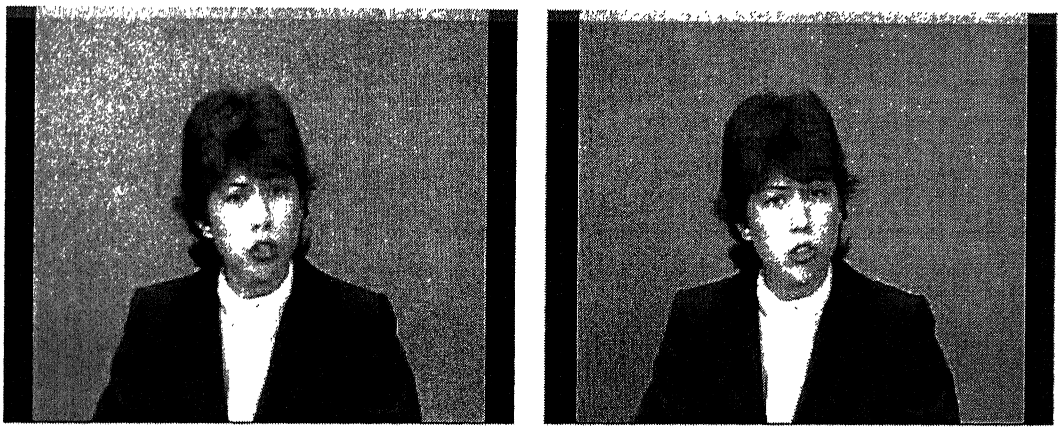
(a) Reconstruction using 'method 1' (b) Reconstruction using 'method 2'

Figure 3.10: 62^{nd} Frame of 'Claire' : Coded at 122Kbps (Compression Ratio 200:1)



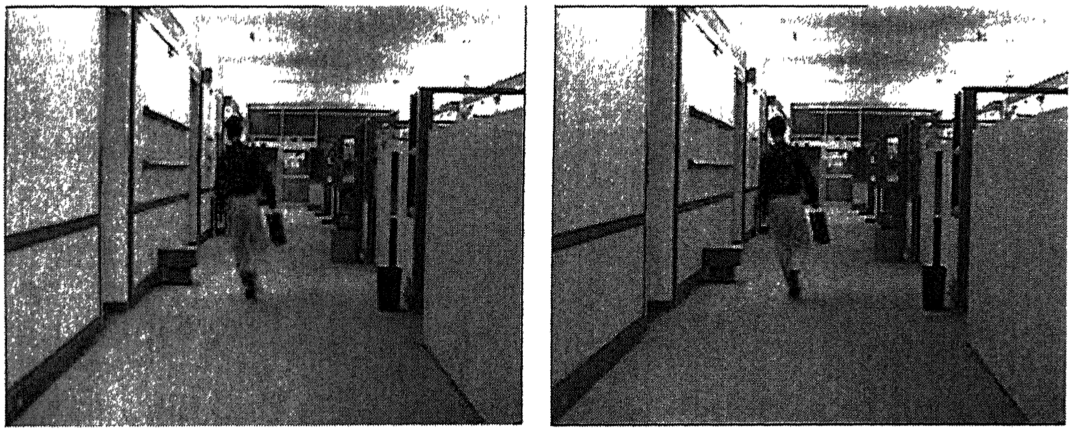
(a) Reconstruction using 'method 1' (b) Reconstruction using 'method 2'

Figure 3.11: 84^{th} Frame of 'Claire' : Coded at 122Kbps (Compression Ratio 200:1)



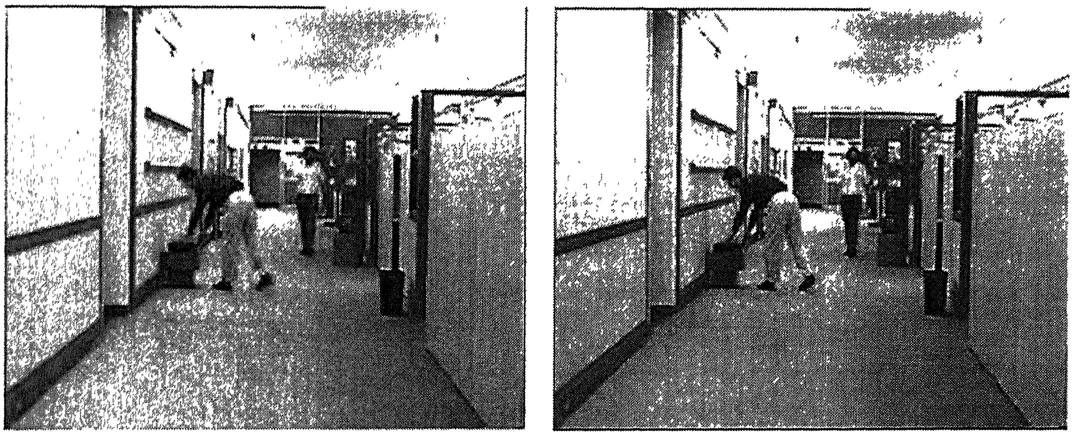
(a) Reconstruction using 'method 1' **(b) Reconstruction using 'method 2'**

Figure 3.12: 136th Frame of 'Claire' : Coded at 122Kbps (Compression Ratio 200:1)



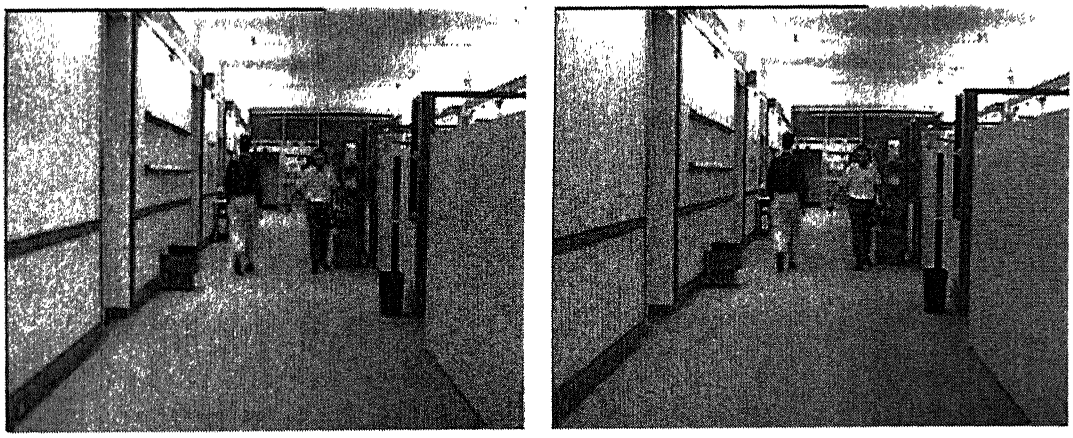
(a) Reconstruction using 'method 1' **(b) Reconstruction using 'method 2'**

Figure 3.13: 66th Frame of 'Hall Monitor' : Coded at 60.8Kbps (Compression Ratio 400:1)



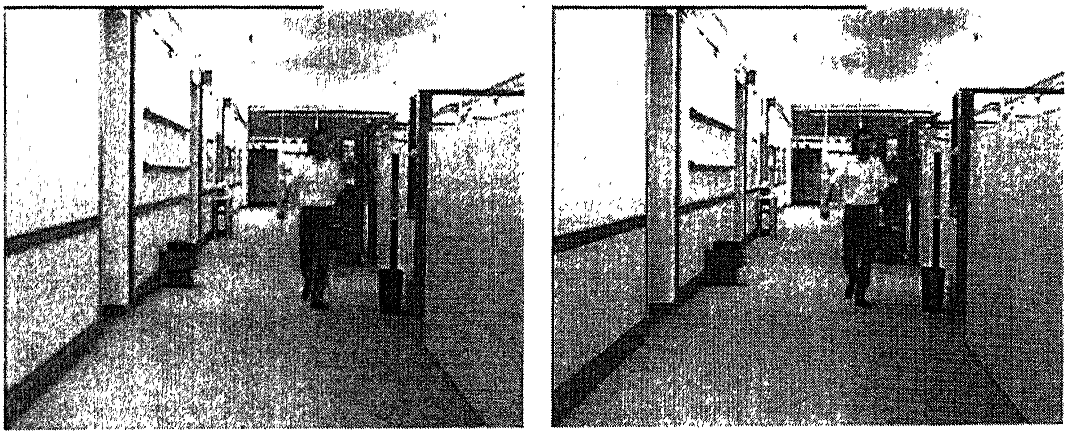
(a) Reconstruction using 'method 1' (b) Reconstruction using 'method 2'

Figure 3.14: 125^{th} Frame of 'Hall Monitor' : Coded at 60.8Kbps (Compression Ratio 400:1)



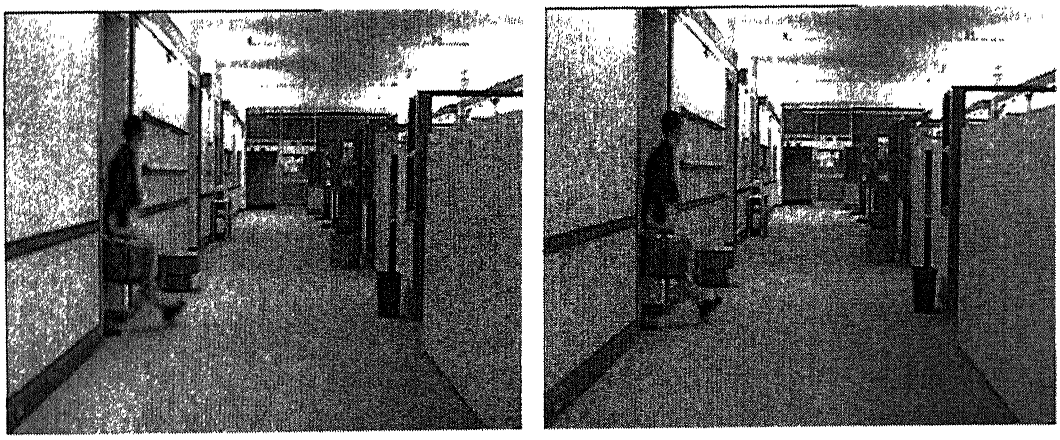
(a) Reconstruction using 'method 1' (b) Reconstruction using 'method 2'

Figure 3.15: 183^{rd} Frame of 'Hall Monitor' : Coded at 60.8Kbps (Compression Ratio 400:1)



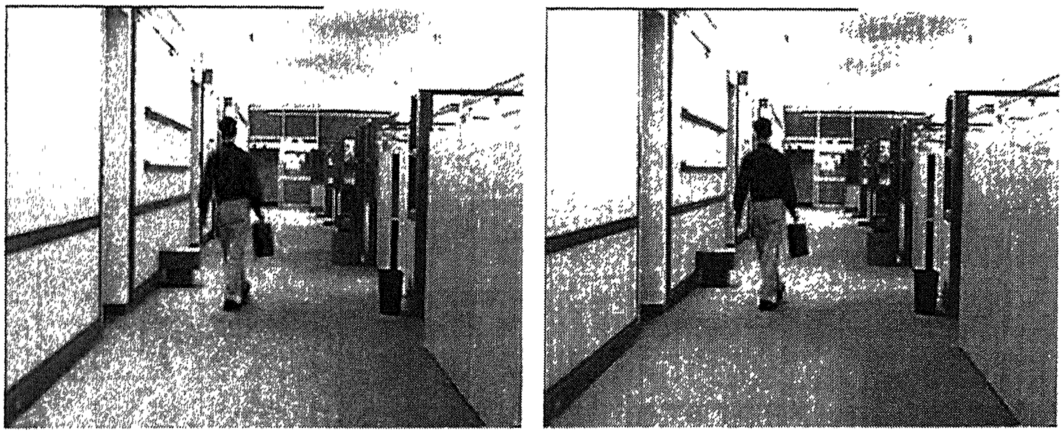
(a) Reconstruction using 'method 1' (b) Reconstruction using 'method 2'

Figure 3.16: 267th Frame of 'Hall Monitor' : Coded at 60.8Kbps (Compression Ratio 400:1)



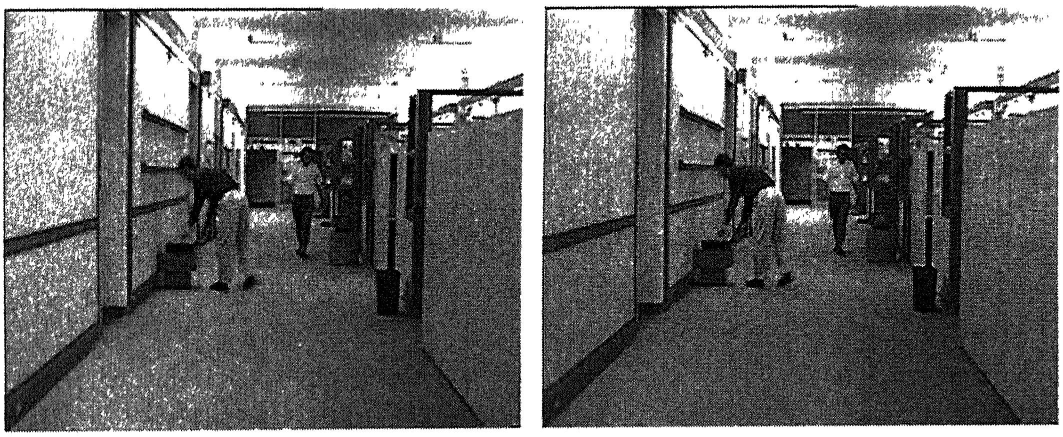
(a) Reconstruction using 'method 1' (b) Reconstruction using 'method 2'

Figure 3.17: 24th Frame of 'Hall Monitor' : Coded at 122Kbps (Compression Ratio 200:1)



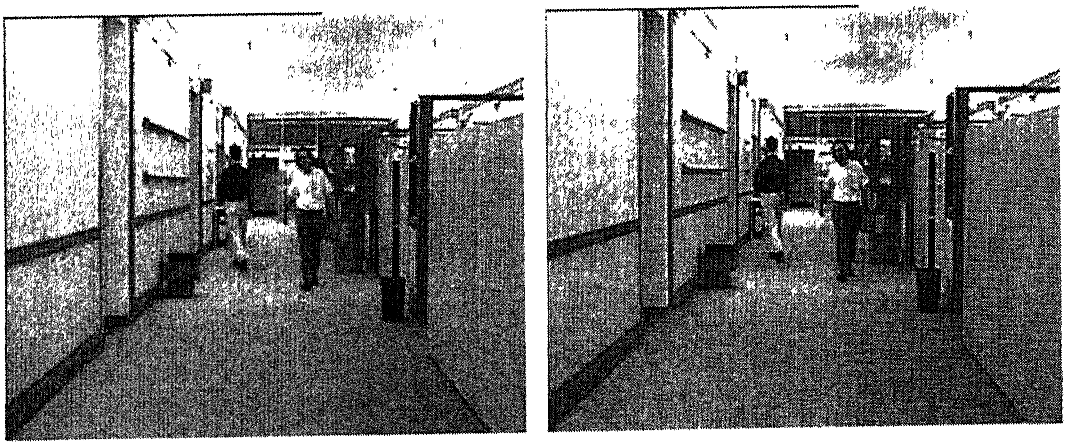
(a) Reconstruction using 'method 1' **(b) Reconstruction using 'method 2'**

Figure 3.18: 57^{th} Frame of 'Hall Monitor' : Coded at 122Kbps (Compression Ratio 200:1)



(a) Reconstruction using 'method 1' **(b) Reconstruction using 'method 2'**

Figure 3.19: 130^{th} Frame of 'Hall Monitor' : Coded at 122Kbps (Compression Ratio 200:1)



(a) Reconstruction using 'method 1' (b) Reconstruction using 'method 2'

Figure 3.20: 214th Frame of 'Hall Monitor' : Coded at 122Kbps (Compression Ratio 200:1)

Chapter 4

Video Coding : A Generalized Approach

4.1 Generalization : why & how?

In general, video contains significant amount of motion information. The motion may arise because of either the object movement or the camera movement (e.g. zooming, panning, rotation) or due to both. Now, presence of motion reduces the smoothness (or in other words, degree of correlation) along the time dimension. This can be verified from Figures 4.1 & 4.2, which show the temporal views of the video sequence, "cartoon". It contains different combinations of complex motions. DCT, in such situation, may not be always a good choice. In fact, DCT will not be able to compress the energy well and the values of the higher frequency coefficients will also remain significant. This would give rise to rapid fluctuation in frame-PSNR value.

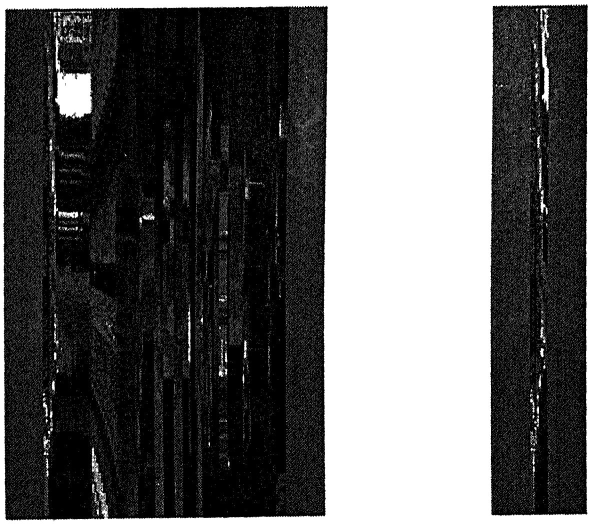
Here, it's worth mentioning that the same problem exists in case of 3D-SBC, where temporal filtering is used instead to decorrelate the data along the time dimension. To tackle this problem, researchers, at first, try to find out the motion trajectories through various motion estimation techniques and then apply the temporal filtering along those trajectories (refer to [5], [6], [9] for further details).

However, in the present case, we avoid motion estimation. Our idea is, if a fixed

Number of frames: 112
(a)

Number of frames: 8
(b)

Figure 4.1: Side View: Temporal Changes Along Vertical Plane



Number of frames: 112
(a)

Number of frames: 8
(b)

Figure 4.2: Vertical View: Temporal Changes Along Horizontal Plane

transform like DCT fails to perform well in any case, why can't we change the transform itself according to the data so as to compensate for the performance loss? Hence, we continue our experiment using some data dependent linear transform (like KLT). This idea may seem somewhat analogous to the use of KLT for spectral decorrelation of multi-spectral images [23]. However, using KLT in the same manner on a video (of course, in this case, for temporal decorrelation) is far more objectionable. The main problem associated with the data dependent linear transform (like KLT) is that depending on the data characteristics the transformation matrix varies widely and hence always needs to be transmitted along with the data (unlike the fixed transforms like DCT) in order to help proper decoding at the decoder end. Obviously, this incurs both larger encoding delay and increased bit-rate. Hence, KLT does not really seem to be a feasible solution.

Still, we have been motivated by the works of few researchers who have recently advocated the use of KLT in the context of universal image compression ([24], [25]) and image representation using the mixture of principal components (MPC) model [26]. In these papers, the transformation matrix is not actually computed for each and every image block (or set of image blocks). Rather, an optimal set of KL transformation matrices is built apriori. During actual image encoding process, image block (or set of image blocks) is transformed using one of these transformation matrices, which closely matches the associated data characteristic. Significant improvement in the rate-distortion performance over the conventional methods has been reported.

In the present context, we have tried to build a similar sub-system that can be used to replace DCT to achieve temporal decorrelation of the GOF. The next section discusses at length this issue.

4.2 Design of codebook

The first requirement for designing a system analogous to those described in ([24], [25], [26]) is the availability of a "codebook", i.e. a set of KL transformation matrices. In this section, we discuss the procedure that we have followed to build the "codebook".

We have used the Generalized Lloyd Algorithm [27] to design the codebook. The procedure is not much different from that described in [26]. In [26], the design starts with a carefully chosen set of transformation matrices such that *no transformation matrix corresponds to the region outside the distribution space of the training data.* A number of methods have been suggested to choose the initial set, e.g.

- arbitrarily partitioning the training data into K classes (where K is the desired number of classes) and estimating the corresponding transformation matrices using batch eigen decomposition,
- using a single fixed-basis transformation such as DCT and adding small amount of random variations to it to produce a set of K unique transformation matrices,
- using an estimate of the global principal components of the entire data and then adding small amount of random variations to produce the initial set of K unique transformation matrices.

With this set of transformation matrices in hand, an iterative approach is followed, where, at each iteration, the training input data is re-classified among previously obtained set of classes based on the minimum mean square error (MMSE) criterion. Also at the end of each iteration, the class-transformation-matrices are updated with the help of Sanger's Generalized Hebbian Algorithm (GHA) [26]. The process is continued until the overall average MMSE drops below a predetermined level or the overall improvement becomes negligible.

4.2.1 Initial codebook design

In the above method, the initial codebook (set of transformation matrices) is chosen in a completely random fashion. This may not always satisfy the requirement that *no transformation matrix should correspond to some region outside the distribution space of the training data.* Also, this sort of arbitrary choice delays the convergence during subsequent iterative updating of codebook. In the following, we suggest a systematic way to build the initial set.

For the sake of this discussion, suppose, we are given a set of three-dimensional vectors $(a_1, b_1, c_1), (a_2, b_2, c_2), \ldots (a_n, b_n, c_n)$. We shall have to form a set of classes out of these training vectors. We do not fix up the number of classes in advance. Because, we feel that this number itself is data dependent.

Now, we start as follows. We take the first data point (a_1, b_1, c_1), consider the line passing through $(0, 0, 0)$ and (a_1, b_1, c_1) as the subspace representing class1. Next, we take the second data point (a_2, b_2, c_2), project the associated vector onto the class1 subspace and measure the Euclidean norms of the projection p and the error e. We define a quantity

$$l = a\,||p|| - \,||e|| \tag{4.1}$$

where $0 \leq a \leq 1$. If

$$l \geq 0 \tag{4.2}$$

we assign the data point (a_2, b_2, c_2) to class1. Otherwise, we construct a new class and take the line passing through $(0, 0, 0)$ and (a_2, b_2, c_2) as the subspace representing the new class.

This procedure continues until the training set is exhausted. Finally, we adjust the subspaces associated to each class by considering all the data point members. For example, we may take the mean of the data points that are assigned to the class under consideration and consider the line passing through the origin and the mean point as the new subspace representing that class.

The same idea may be extended to the case where each of the training data is not a mere point, rather a set of points. For example, in case of a three-dimensional pixel-block of size $d_1 \times d_2 \times d_3$ we can think of $d_1 d_2$ number of d_3 dimensional data-points/vectors (refer to Figure 4.3). For the time being, say, $d_3 = 3$. Now, we can form the subspaces in the following manner. We compute the correlation matrix C corresponding to the first set of points as

$$C_{d_3 \times d_3} = \frac{1}{d_1 d_2} \sum_{i=1}^{d_1 d_2} x_{d_3 \times 1} x^T_{d_3 \times 1} \tag{4.3}$$

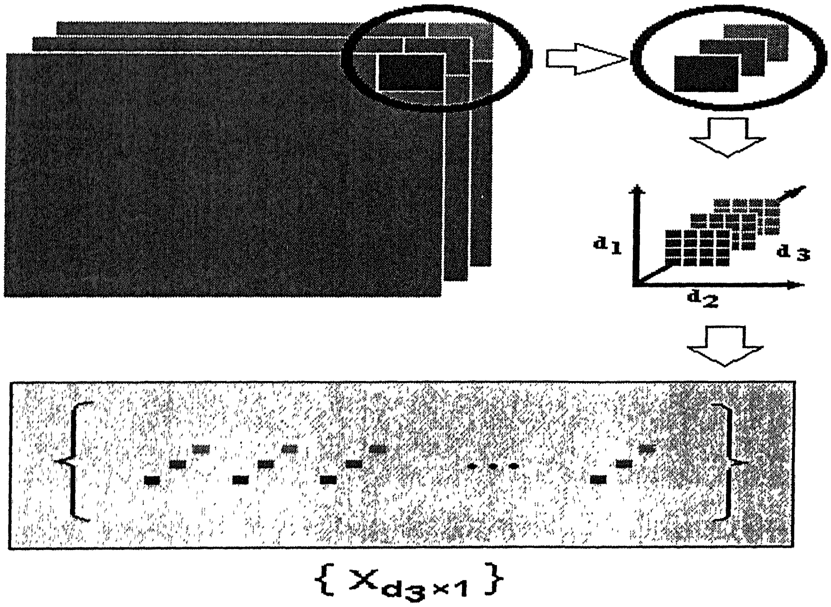


Figure 4.3: d_1d_2 number of d_3 dimensional data-points

We find out the normalized eigen vectors of the correlation matrix and arrange them row-by-row into another matrix $[F]_{d_3\times d_3}$ such that the first row corresponds to the largest eigen value, the second one corresponds to the second largest eigen value and so on. Next, let us make the elements in the last row of $[F]_{d_3\times d_3}$ all zero and form another matrix $\left[\hat{F}\right]_{d_3\times d_3}$. Note that

$$\left[\hat{F}\right]_{d_3\times d_3} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} [F]_{d_3\times d_3} \tag{4.4}$$

and the rows of $[F]_{d_3\times d_3}$ *and* $\left[\hat{F}\right]_{d_3\times d_3}$ form an orthonormal set. Now, if we define the subspace as the *span* of the rows of $\left[\hat{F}\right]_{d_3\times d_3}$, then we can also define the projection p of an arbitrary vector $V = [v_1, v_2, v_3]^T$ on this subspace and the corresponding error e as

$$p = \left[\hat{F}\right]^T \left[\hat{F}\right] V \tag{4.5}$$

$$e = V - p \tag{4.6}$$

In the process of classifying the entire set of the set of points, we consider the individual sets one at a time, extract some useful feature (say, the mean) out of it, project it (i.e. the mean) on the existing subspaces, select the subspace that is able to represent it with minimum error. If this minimum error satisfies Equations (4.1) & (4.2), the set under consideration is identified as the member of that subspace. Otherwise, a new subspace is formed. Repeating this sequence of operations for all the available sets of points, we ultimately end up with a set of classes/subspaces. Each class is represented by a KL transformation matrix.

4.2.2 Iterative codebook updating

Once the initial set is ready, we follow the LBG algorithm [27] to obtain the final "codebook". The iterative form of the LBG algorithm can be put as follows:

1. Initialization : Given an initial set of K transformation matrices, W_0, where $W_0 = \{[F]_{01}, [F]_{02}, \dots [F]_{0K}\}$, a training data-set $\{X_j \; ; \; j = 1 \dots N\}$, a distortion measure e as in Equation (4.6), a distortion threshold $\epsilon \geq 0$, set $m = 0$ and $D_{-1} = \infty(i.e.\ a\ large\ value)$.

2. Given $W_m = \{[F]_{m1}, [F]_{m2}, \dots [F]_{mK}\}$, find the minimum distortion partition $P(W_m) = \{S_i \; ; \; i = 1 \dots K\}$ of the training sequence such that $X_j \in S_i\ if\ e_{ji} \leq e_{jl}\ for\ all\ values\ of\ l$. Compute the average distortion

$$D_m = \frac{1}{N} \sum_{j=1}^{N} \min_{k \in \{1, \dots K\}} e_{jk} \tag{4.7}$$

3. If $(D_{m-1} - D_m) \leq \epsilon D_m$, halt with W_m, otherwise continue.

4. Update the set W_m by calculating the eigen vectors of the class correlation matrices once again. Set $W_{m+1} = W_m$. Replace m by $m + 1$ and go to step (2).

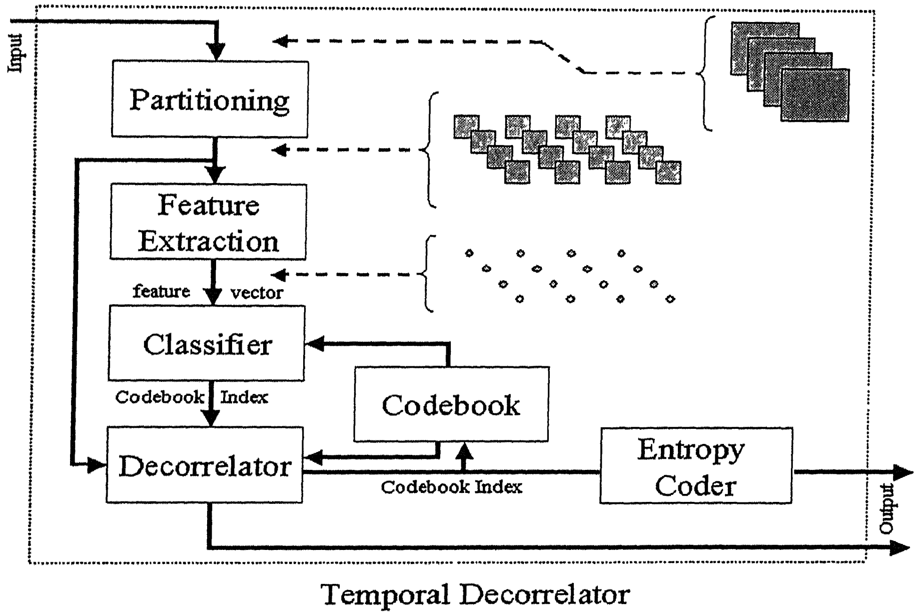


Figure 4.4: Temporal Decorrelator in the Generalized Encoder

4.3 General structure of the proposed codec

In the general structure of the proposed codec, we replace Discrete Cosine Transform with a set of KL Transforms. Figure 4.4 shows the internal block diagram of the Temporal Decorrelator. The GOF is partitioned into three-dimensional data blocks of size (say) $S \times S \times L$. For CIF frame-resolution-format, we have taken $S = 8$. L is the number of frames in GOF, and is equal to 8 in our case. Each data block is passed through the "Feature Extractor". The "Feature Extractor" extracts some useful feature out of that data block. In our case, the feature is nothing but the mean vector $[M]_{L\times 1}$, where

$$[M]_{L\times 1} = \frac{1}{S \times S} \sum_{i=1}^{S\times S} [x_i]_{L\times 1} \tag{4.8}$$

To make this mean vector a useful representative of the data block, we should not take very large value of S. However, the value of S itself can be varied depending on the region. For example, we can take larger value of S for the regions having little spatial change. But, for simplicity, we have kept S fixed. The feature is then passed to the "Classifier". The "Classifier" finds out the class/subspace/KL-transform that can best represent the input feature. In fact, it projects the mean vector on the subspaces spanned by the *first four rows* of each of the KLTs. The one that gives the lowest projection error is ultimately selected. The chosen KLT matrix is next used by the "Decorrelator" to decorrelate the data block along the time dimension. The corresponding class-index is subsequently entropy coded and sent as an additional data to the decoder.

At the decoder end, the class-index is used to know the exact KLT that has been used for a particular data block.

4.4 Simulation result

Experiment has been carried out on a high-motion-test-video "Irene". This video has CIF (352 × 288) frame resolution with frame rate of 10 frames/sec. In this video, background is always stationary. But, there are rapid movements of fingers, hands, face and other body-parts of the person standing in the foreground. As a training data, we have used the first half duration of this video. Codebook thus generated has been applied to encode the entire video. Figure 4.5 shows the variation of frame-PSNR as obtained using this generalized codec (let us call it 'codec 2') and the codec proposed in the second chapter (call it 'codec 1'). Both of them are used to encode the video at $0.3bpp$ or equivalently at $304Kbps$. The comparison shows that the fluctuation in the frame-PSNR is reduced with 'codec 2'.

Finally, Figures 4.7, 4.8, 4.9 and 4.10 show some of the results obtained using 'codec 2' at bitrates of $304Kbps$, $506Kbps$, $1Mbps$ and $2Mbps$ respectively.

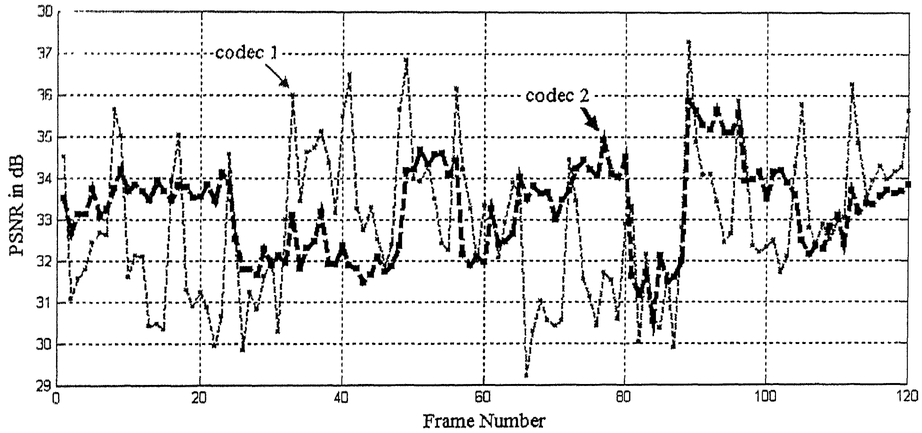


Figure 4.5: Performance Comparison Between 'codec 1' and 'codec 2'

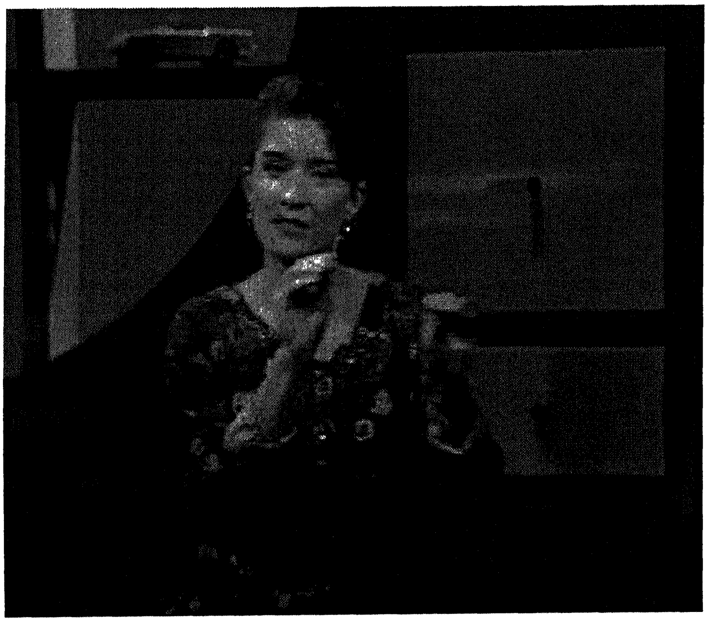


Figure 4.6: Original 73^{rd} Frame of 'Irene' Sequence

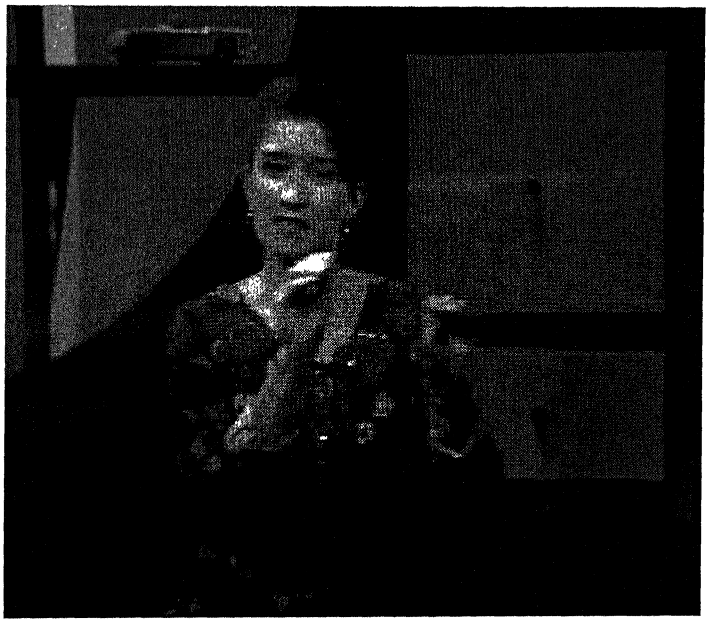


Figure 4.7: 73^{rd} Frame of 'Irene' : Coded at 304Kbps

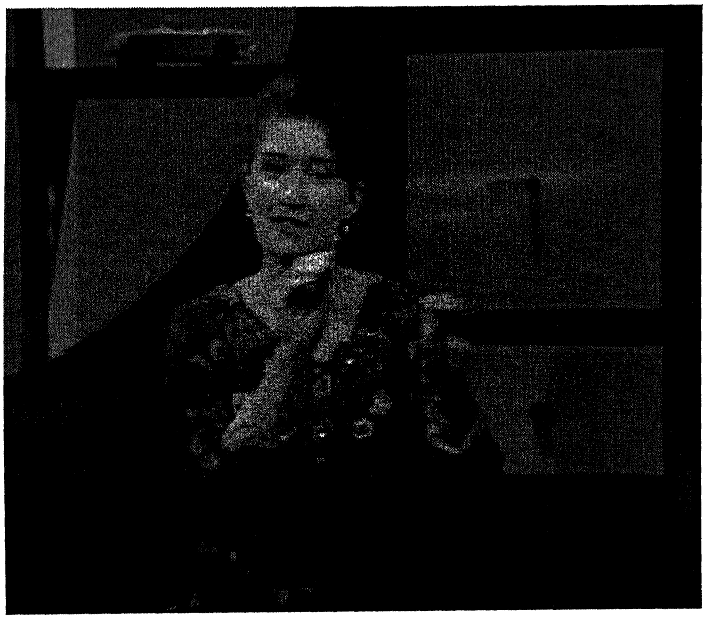


Figure 4.8: 73^{rd} Frame of 'Irene' : Coded at 506Kbps

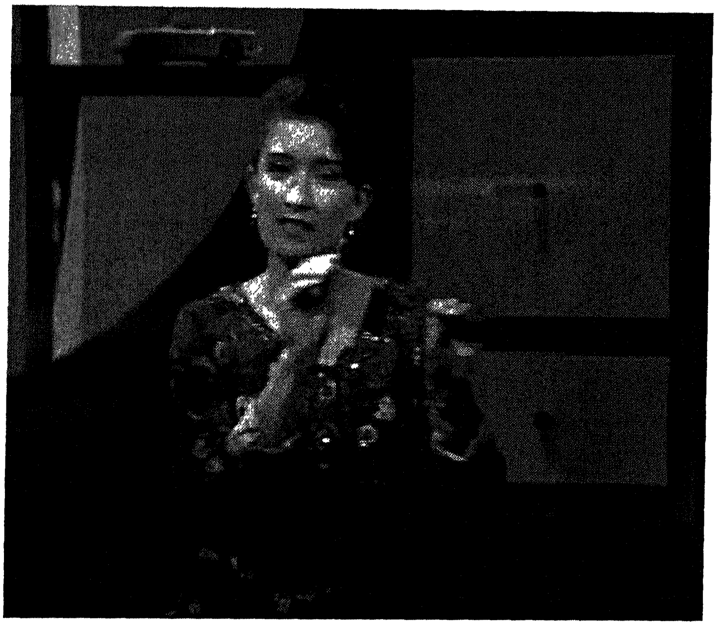


Figure 4.9: 73^{rd} Frame of 'Irene' : Coded at 1Mbps

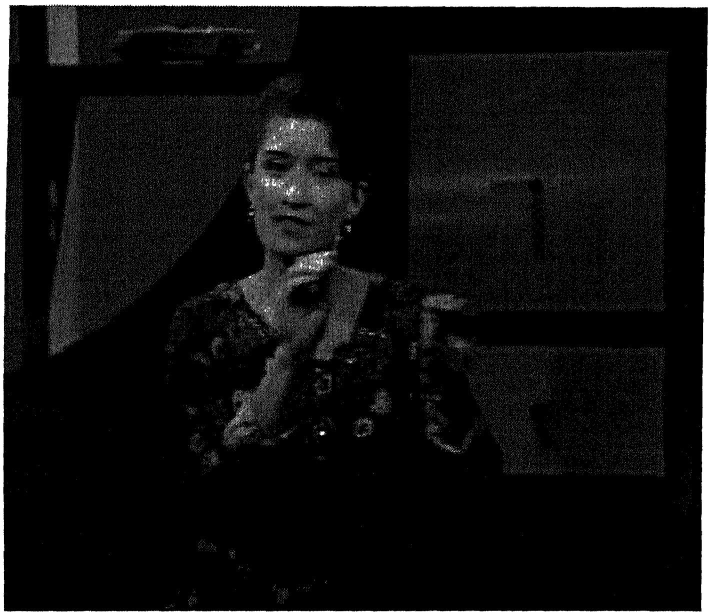


Figure 4.10: 73^{rd} Frame of 'Irene' : Coded at 2Mbps

Chapter 5

Scalable Coding: A Much Desired Feature

Now-a-days, video-coding is not only confined to compressing spatio-temporal visual information but is also focussed to incorporate additional features like scalability, region-of-interest coding etc. Moreover, increasing demand of video-delivery over Internet, unreliable wireless-mobile channel has necessitated restructuring of the entire coding strategy so as to produce a bit-stream sufficiently robust against packet loss and burst error type transmission impairments. In this chapter, we shall discuss only the scalability aspect. We will highlight on how scalable coding can be achieved using the proposed codec and leave the issues like region-of-interest coding, efficient channel coding as a part of future-investigation.

5.1 Scalability: why is it needed?

In the last few decades we have seen the tremendous growth of both Internet and multimedia technologies. Day by day, it is giving birth to new interesting and useful applications and services. Video-On-Demand (VOD), Interactive TV (ITV), Interactive Hypermedia Courseware is just a few among them. Let us consider the Video-On-Demand application for our discussion. There must be a multimedia-video server

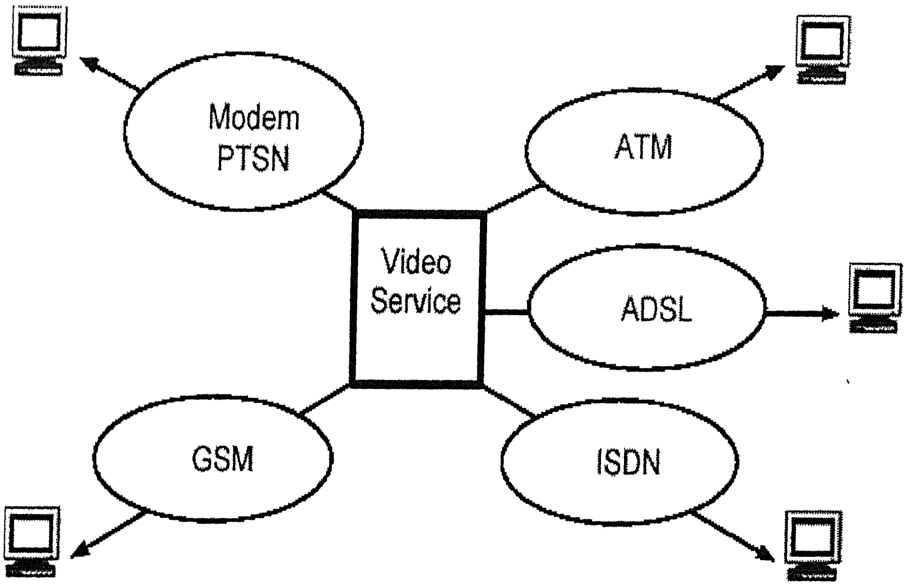


Figure 5.1: A Heterogeneous Communication Network with Video Services.

where all the videos are stored on-line, that is, on magnetic disk. Also there will be thousands of clients/users connected to it through different networks, Internet, demanding for videos. Now, it is quite natural that these users will have wide varieties of computational resource, display and memory capability and screen resolution. Also the available network bandwidth may differ from client to client, time to time. Network reliability (e.g. packet loss characteristics in case of Internet) may fluctuate in random fashion. In such a receiver-driven multicast application in a complicated heterogeneous environment, in order to guarantee the Quality-of-Service (QoS) to all the users, the server needs to optimize the video streaming for each of the individual user! However this task is rather involving and almost impossible to carry out. As an alternative solution, the server can serve one common omnipotent video stream that can simultaneously support all the necessary requirements tailored to individual services.

Scalable video coding, sometimes also called Layered video coding is an approach to materialize the second elegant solution. A conventional method to attain scalability is to split and distribute the video information over a number of layers. The most fundamental information to reconstruct the video frames is packed into a layer referred to as base layer. The other layers, called enhancement layers, are for additional information which, when combined with the base or lower enhancement layers, produce more refined information. More the number of layers the decoder receives, the better is the quality it will achieve. To date, standards like H.263, MPEG-2, and MPEG-4 have accepted this method.

A comparatively newer method to support scalability is to produce an embedded bit-stream. An embedded bit-stream can be truncated from the end to match any reduced target bit-rate. Any truncated version of the parent bit-stream can reconstruct back the video. However, the actual quality of the decoded video depends on the point of truncation. The closer is the point toward the end, the better is the quality. Noteworthy, much of this work was inspired by the design and excellent coding efficiency of the EZW and SPIHT coders for still images. Bits are packed in the order of "significance", a measure that is often tied to the magnitude of image coefficients in the transform domain. Therefore, the image coefficients are ordered first by their magnitude and then packed in a bit-stream in descending order of bit-planes. In EZW and SPIHT the embedding is so fine granular that one can scale the size of a bit-stream in byte accuracy. For this reason, an embedded bit-stream is said to have fine-granularity scalability (FGS). Embedding a video bit-stream is a simple straightforward extension of embedded image coding if a 3-D video transform is used.

Structuring the video bit stream in this manner helps the individual receiver to select some part of the bit stream and decode it with its available capabilities to fulfill its own requirement. Also ideally, the different subset bit-streams extracted from the full parent bit-stream provide the maximum possible amount of video information to any user despite its limited computing power, connection bandwidth, and so on. In this way, by indirectly adapting the content made for some high-end machines to some less powerful machines, the overheads involved in producing different versions of the same information for different clients can be virtually eliminated.

5.2 Scalability: its different forms

There may be many different types of scalability, e.g. temporal resolution (frame-rate) scalability, spatial resolution scalability, bit-rate (SNR) scalability, transmission scalability, and object scalability, to name a few. A system that can handle all of these different scalability aspects would likely to be of great importance in near future. However, in our present discussion, we will confine ourselves within three major kinds of scalability: temporal resolution scalability, spatial resolution scalability, and data-rate scalability.

5.2.1 Temporal Resolution Scalability

Temporal resolution (frame-rate) scalability allows the user to adjust the refresh rate of the decoded frames while playing back the video.

In standard hybrid video coding system e.g. in all MPEG family video coding standards and H.263+, temporal scalability is achieved by including B-frames in a compressed video bit-stream. B-frames are bi-directionally predicted from the forward and the backward I- or P-frames, and not used as reference pictures for motion compensation in reconstructing other B- or P-frames. For this reason, a B-frame can be dropped without affecting the picture quality of other frames. However, dropping a B-frame will lower the frame rate or temporal quality of a video playback, giving the meaning of temporal scalability.

In 3D video coding system, however, temporal scalability is almost an inherent feature. If lesser number of temporal subbands are used at the decoder side to reconstruct back the video, we obtain the lower temporal resolution version of the original video. And if we use more number of temporal subbands, we can reconstruct the higher temporal resolution video.

5.2.2 Spatial Resolution Scalability

Spatial resolution scalability is a functionality to decode frames at different sizes, e.g., from size of thumbnail for fast browsing to that of HDTV.

Spatial scalability is obtained by creating a multiresolution representation of each frame in a video bit-stream. This multiresolution representation is used to split each frame into set of layers. In this case an increased number of reconstruction layers correspond to higher spatial resolution of the individual frames of the video. It is worth noting that an enhancement layer can also be split into multiple enhancement layers so that each one is built upon the previous ones at a lower resolution.

In MPEG, H.263+, the multiresolution representation is attained in DCT frame-work. Whereas, in case of 3D video coding system, the same is obtained in subband-coding/wavelet frame-work.

5.2.3 Data-rate Scalability

Data-rate (SNR) scalability helps to attain progressively increasing quality as more and more of the bit-stream is decoded. This particular type of scalability applies to most types of media like video, sound, speech etc. It is generally achieved using layered quantization technique that splits the bit-plane depth into a set of layers.

Finally, it's worth mentioning, a scalable bit-stream does not always have a single type of scalability. In fact, different types of scalability often co-exist in a multi-dimensional structure, so as to provide a wide range of adaptation choices.

5.3 Potential of the proposed method

Having a short discussion on major three types of scalability in the previous section, let us now see how those features can be provided with the proposed video-codec. In this section, we suggest the ways to incorporate the features in our coding scheme. However, a comprehensive technical solution is beyond our present scope. But, we hope this may serve as a starting point of any future endeavour and further investigations.

In order to understand how exactly we can achieve the aforesaid three types of scalability, let us first restructure our video bit-stream. In section 2.5, we have roughly talked about the hierarchical two-layer bit-stream structure. Let us now introduce one new layer in addition to the "group of frequency-planes layer" and the "frequency-

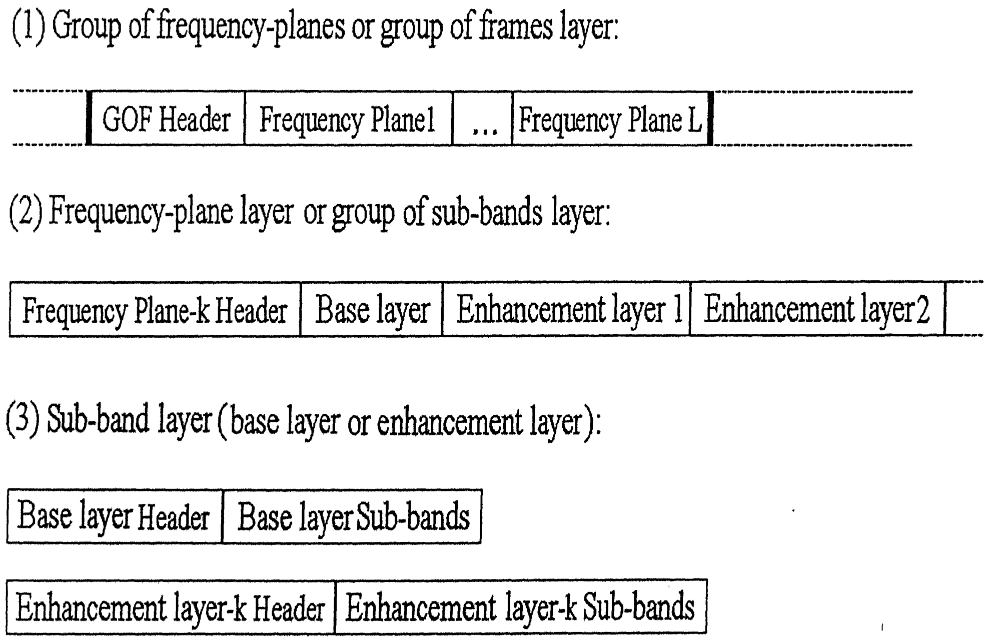

(1) Group of frequency-planes or group of frames layer:
GOF Header
Frequency Plane1
...
Frequency Plane L
(2) Frequency-plane layer or group of sub-bands layer:
Frequency Plane-k Header
Base layer
Enhancement layer 1
Enhancement layer2
(3) Sub-band layer (base layer or enhancement layer):
Base layer Header
Base layer Sub-bands
Enhancement layer-k Header
Enhancement layer-k Sub-bands

Figure 5.2: The Modified Bit-stream Structure

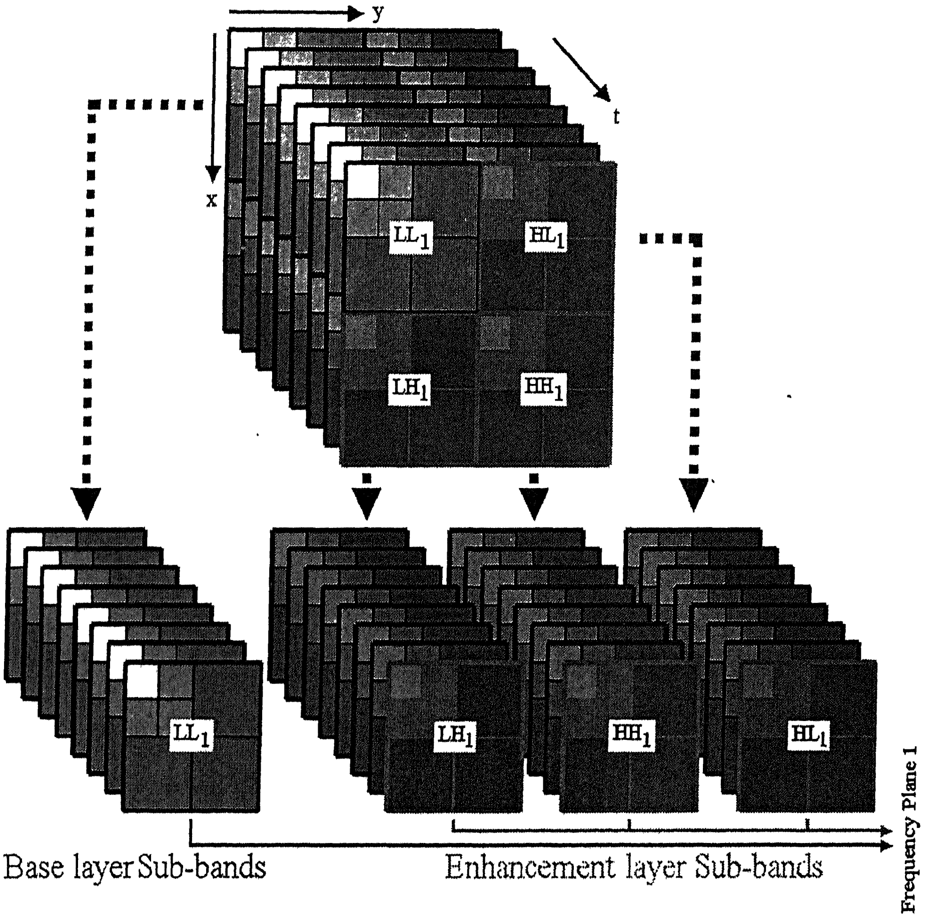


Figure 5.3: Subband Decomposition Required To Achieve The Modified Bit-Stream Structure

plane layer". Let us call it "sub-band layer". The bit stream structure with these three layers has been shown in Figure 5.2. At present, let us assume that the number of "enhancement layer"s in each frequency-plane be one. Now, to construct the bit stream corresponding to each of the layers, we shall have to decompose the group of frequency-planes in a manner shown in Figure 5.3. Here, for each frequency plane LL_1 corresponds to the base layer and the remaining subbands i.e. LH_1, HL_1 *and* HH_1 constitute the single enhancement layer. We apply DWT and SPIHT coding to each of these subbands to generate the corresponding bit-streams.

Next, let us consider the following three distinct choices that a decoder can have while reconstructing back the video.

1. The decoder can choose to use the full bit-stream pertaining to a particular frequency-plane, but select only first few of the available frequency planes while decoding.
2. The decoder can pick up all the frequency planes, but use only the base layers to construct the video.
3. The decoder can pick up all the frequency planes, take all the enhancement layers in addition to the base layers, but use the truncated bit-stream corresponding to each layer.

It is interesting to note that in the first case, the decoder will reconstruct a lower temporal resolution video. The more is the number of selected frequency planes, the better is the temporal resolution achieved. The idea is quite interesting. Because, here we do consider each frequency plane as a separate temporal subband and attempt to incorporate temporal scalability in a manner that 3D-SBC has shown in the past. In the second case, however, the reconstructed video will be of lower spatial resolution. If the decoder would have used the enhancement layer too, it could have had the full spatial resolution. And finally, the third way of decoding will simply give rise to a low quality video. Had the decoder used more number of bits from each bit-stream layer, the decoded quality could have been improved to the corresponding extent.

Therefore, by organizing the video bit-stream as shown in Figure 5.2, we can easily include the scalability feature in our coding scheme. However, unlike the way we have just discussed, in actual scenario, any decoder will have a mixture of all these experiences and different decoders will be able to decode the video to satisfy individual requirements.

Chapter 6

Conclusion And Future Scope

In this thesis work, we have tried to open up some alternative approaches that can potentially build a mature video-codec suitable for the wide range of video-compression applications.

At first, we have proposed a very low bit rate 3D video coding scheme. This coding method can be considered as a hybrid of 3D-DCT and 3D-SBC techniques. DCT is used to exploit the temporal redundancy and SBC (wavelet coding) is used to exploit the spatial redundancy present in the video signal. The proposed codec is used to code low-motion videos (with frame format: CIF (352×288), frame rate: 30 frames/sec) at bitrate around 120Kbps. The subjective quality of the decoded video is satisfactory. We are also able to attain the desired bit-rate accurately.

However, the quality of the reconstructed video-frame deteriorates when the motion activity as a whole increases. Also at even lower bit-rate (e.g. around 60Kbps), significant distortion is noticed at the regions having larger motion. Next, we look into this problem and suggest a modification on the previous algorithm in order to prevent this degradation in quality. The modified algorithm tries to identify the regions having large motion and allocates more bits to code these regions. Experimental results have been shown to demonstrate the improvement achieved in the decoded video quality. It is worth mentioning that using this modified coding strategy, we are able to reduce the bit-rate below 64Kbps.

Subsequently, we try to generalize the proposed coding method. DCT is replaced with a set of KLTs in order to cope up with the varying temporal nature of any general video. This KLT-set is generated apriori using some iterative training algorithm. The proposed method can be considered as some alternative to the traditional motion compensation technique that is used in combination with 3D-SBC in order to reduce the fluctuation in frame-PSNR in high-motion video coding scenario. Simulation results are given to support the view.

Finally, we discuss the scalability issue. We intimate how scalability can be achieved using the proposed codec.

Throughout the work, not much attention was paid to optimize the proposed algorithms. The primary goal was to see that the ideas proposed in this thesis indeed work. Rigorous optimization of different parts of these algorithms can be taken up as future work. In addition, thorough investigation of the generalized scheme should be made with an exhaustive codebook built out of a carefully chosen set of training video. Finally, we have already mentioned that the proposed codec has the potential to achieve scalable, error-resilient, region-of-interest video coding. We have also discussed the ways to incorporate some of those features. Further research is still required to design a full-featured video codec based on the proposed scheme.

Bibliography

[1] Yui-Lam Chan and Wan-Chi Siu, "Variable Temporal-Length 3-D Discrete Cosine Transform Coding," *IEEE Transactions on Image Processing*, vol. 6, No. 5, pp. 758-763, May 1997.

[2] G. Karlsson and M. Vetterli, "Three dimensional subband coding of video," *Proc. Int. Conf. Acoustics, Speech, Signal Processing (ICASSP)*, pp. 1100-1103, April 1988.

[3] C. I. Podilchuk, N. S. Jayant and N. Farvardin, "Three-dimensional subband coding of video," *IEEE Transactions on Image Processing*, vol. 4, No. 2, pp. 125-139, February 1995.

[4] T. Kronander, "New results on 3-dimensional motion compensated subband coding," *Proc. Picture Coding Symposium (PCS90)*, March 1990.

[5] J. R. Ohm, "Three-dimensional subband coding with motion compensation," *IEEE Transactions on Image Processing*, vol. 3, No. 5, pp. 559-571, September 1994.

[6] S. J. Choi and J. W.Woods, "Motion-compensated 3-D subband coding of video," *IEEE Transactions on Image Processing*, vol. 8, No. 2, pp. 155-167, February 1999.

[7] V. M. Bove and A. B. Lippman, "Scalable open-architecture television," *SMPTE (Society of Motion Picture and Television Engineers) Journal*, pp. 2-5, January 1992.

[8] D. Taubman and A. Zakhor, "Multirate 3-D subband coding of video," *IEEE Transactions on Image Processing*, vol. 3, No. 5, pp. 572-588, September 1994.

[9] B.-J. Kim, Z. Xiong and W. A. Pearlman, "Low bit-rate scalable video coding with 3D set partitioning in hierarchical trees (3D SPIHT)," *IEEE Transactions on Circuits & Systyms for Video Technology*, vol. 10, No. 8, pp. 1365-1374, December 2000.

[10] A. Said and W. A. Pearlman, "A new fast and efficient image codec based on set partitioning in hierarchical trees," *IEEE Transactions on Image Processing*, vol. 5, pp. 1303-1310, September 1996.

[11] A. Said and W. A. Pearlman, "Image compression using the spatial-orientation tree," *Proc. IEEE Int. Symp. Circuits and Systems*, May 1993, pp. 279-282.

[12] J . M. Shapiro, "Embedded image coding using zerotrees of wavelet coefficients," *IEEE Transactions on Signal Processing*, vol. 41, pp. 3445-3462, December 1993.

[13] Y. Chen and W. A. Pearlman, "Three-dimensional subband coding of video using the zero-tree method," *Proc. SPIE 2727Visual Communications and Image Processing 96*, pp. 1302-1309, March 1996.

[14] J. Y. Tham, S. Ranganath and A. A. Kassim, "Highly scalable waveletbased video codec for very low bit-rate environment," *IEEE J. Select. Areas Commun.*, vol. 16, pp. 12-27, January 1998.

[15] Sungdae Cho and William A. Pearlman, "A Full-Featured, Error-Resilient, Scalable Wavelet Video Codec Based on the Set Partitioning in Hierarchical Trees (SPIHT) Algorithm," *IEEE Transactions on circuits and systems for video technology*, Vol. 12, No. 3, pp. 157-171, March 2002.

[16] ITU SG 16 Q.15, "H.26L Test Model Long Term Number 3 (TML-3)," Doc. Q15J08, Osaka, Japan, May 2000.

[17] Khalid Sayood, "Introduction to Data Compression," 2nd Edition, Academic Press, Morgan Kaufmann Publishers, 2000.

[18] Rafael C. Gonzalez and Richard E. Woods, "Digital Image Processing," Delhi, 2002, 2nd Edition, Pearson Education Asia.

[19] Anil K. Jain, "Fundamentals of Digital Image Processing," New Delhi, 2001, Prentice-Hall of India.

[20] Vasudev Bhaskaran and Konstantinos Konstantinides, "Image and Video Compression Standards *Algorithms and Architectures*," Dordrecht, 1995, Kluwer Academic Publishers.

[21] Atul Puri and Tsuhan Chen, "Multimedia Systems, Standards, and Networks," New York, 2000, Marcel Dekker Inc.

[22] Bryan E. Usevitch, "A Tutorial on Modern Lossy Wavelet Image Compression: Foundations of JPEG 2000," *IEEE Signal Processing Magazine*, Vol. 18, pp. 22-35, September 2001.

[23] John A. Saghri, Andrew G. Tescher and John T. Reagan, "Practical Transform Coding of Multi-Spectral Imagery," *IEEE Signal Processing Magazine*, Vol. 12, pp. 32-43, January 1995.

[24] Michelle Effros, "Optimal Modeling for Complex System Design [Data Compression]," *IEEE Signal Processing Magazine*, Vol. 15, pp. 51-73, November 1998.

[25] Michelle Effros, Philip A. Chou and Robert M. Gray, "Weighted Universal Image Compression," *IEEE Transactions on Image Processing*, Vol. 8, pp. 1317-1329, October 1999.

[26] R. Donny and S. Haykin, "Optimally Adaptive Transform Coding," *IEEE Transactions on Image Processing*, Vol. 4, pp. 1358-1370, October 1995.

[27] Y. Linde, A. Buzo, R. M. Gray, "An Algorithm for Vector Quantizer Design," *IEEE Transactions on Communications*, Vol. COM-28, No. 1, pp. 84-95, January 1980.

[28] Mika Helsingius, Pauli Kuosmanen and Jaakko Astola, "Image Compression using Multiple Transform," *Signal Processing: Image Communication*, pp. 513-529, 15 (2000), Elsevier Science B.V.

[29] B. B. Chai, J. Vass and X. Zhuang, "Significance-linked connected component analysis for wavelet image coding," *IEEE Transactions on Image Processing*, Vol. 8, No. 6, pp. 774-784, June 1999.

[30] D. Taubman, "High performance scalable image compression with EBCOT," *IEEE Transactions on Image Processing*, Vol. 9, pp. 1158-1170, July 2000.

A144427

A144427